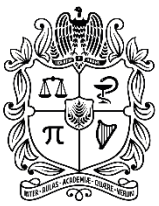


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Metodología de Diseño de Filtros Digitales Basada en Estrategias de Optimización Metaheurística

Fabio Andrés Ramírez Quiroz

Universidad Nacional de Colombia

Facultad de Minas, Departamento de Energía Eléctrica y Automática

Medellín, Colombia

2017

Metodología de Diseño de Filtros Digitales Basada en Estrategias de Optimización Metaheurística

Fabio Andrés Ramírez Quiroz

Tesis de investigación presentada como requisito parcial para optar al título de:

Magister en Ingeniería - Ingeniería Eléctrica

Director:

Ph.D., Freddy Bolaños Martínez

Codirectora:

Ph.D. Eliana Isabel Arango Zuluaga

Línea de Investigación:

Inteligencia Computacional y Procesamiento de Señales

Grupo de Investigación:

Grupo de Automática de la Universidad Nacional GAUNAL

Universidad Nacional de Colombia

Facultad de Minas, Departamento de Energía Eléctrica y Automática

Medellín, Colombia

2017

Resumen

En el presente documento se exponen los resultados y productos de la tesis de investigación necesaria para la acreditación del título de Magister en Ingeniería – Ingeniería Eléctrica, otorgado por la Universidad Nacional de Colombia. La investigación estuvo dirigida a proponer, desarrollar e implementar un método de diseño de filtros digitales lineales e invariantes en el tiempo, FIR (*Finite Impulse Response*) e IIR (*Infinite Impulse Response*), que considere la cuantificación de los coeficientes del filtro desde el comienzo del proceso de diseño, y no al final, como ocurre en otras metodologías, incluidas las metodologías tradicionales, en las cuales, dicha acción puede incrementar el error de aproximación al filtro deseado, hasta la inestabilidad.

Las metodologías de diseño propuestas se basan en los algoritmos de optimización metaheurística APBIL (*Adaptive Population-Based Incremental Learning*), GA (*Genetic Algorithm*) y TS (*Tabu Search*), y están orientadas a obtener la combinación de los coeficientes que mejor se aproxime a la respuesta en frecuencia deseada, para las representaciones binarias de punto fijo (complemento a dos) y punto flotante (según el estándar IEEE 754). Los resultados de las simulaciones son prometedores y muestran un buen desempeño de los filtros obtenidos (destacando el uso del algoritmo APBIL), en términos del Error Medio Cuadrático entre las respuestas en frecuencia deseada y la obtenida a partir de filtro optimizado. Como resultado, se construyó un *software* amigable, capaz de proporcionar diseños competentes, que logran superar en calidad a los diseños tradicionales, especialmente si el número de coeficientes y/o la precisión numérica son limitados.

Palabras clave: Aprendizaje Incremental Basado en Poblaciones, Algoritmos Genéticos, Búsqueda Tabú, diseño óptimo, filtros digitales, cuantificación, procesamiento digital de señales.

Abstract

This document presents the results and products of the research thesis for the accreditation of the Magister's degree in Engineering – Electrical Engineering, granted by the National University of Colombia. The research was aimed at proposing, developing and implementing a method of designing linear and time-invariant digital filters, FIR (Finite Impulse Response) and IIR (Infinite Impulse Response), which considers the quantification of digital filter coefficients since the beginning of the design process, and not at the end, as in others methodologies, including the traditional methodologies, in them this action might increase the approach error to the desired filter, until the instability.

The proposed design methodologies are based on APBIL (Adaptive Population-Based Incremental Learning), GA (Genetic Algorithm) and TS (Tabu Search) metaheurisitc optimization algorithms, and are oriented to obtain the combination of coefficients that best approximates the response at desired frequency, for fixed point (two's complement) and floating point (according to Std IEEE 754) binary representations. The results of the simulations are promising and show a good performance of the obtained filters (highlighting the use of the APBIL algorithm), in terms of the Mean Squared Error between the desired frequency response and that obtained from the optimized filter. As a result, a user-friendly software has been built to provide competent designs that are able to outperform traditional designs, especially when number of coefficients and/or numerical precision are limited.

Keywords: Population-Based Incremental Learning, Genetic Algorithm, Tabu Search, optimal design, digital filter, quantification, digital processing signal.

Contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

Introducción

En el momento de su aparición, en la década de 1950, los filtros digitales nacen como modelos de simulación de los filtros analógicos de la época, a partir de lo cual, las primeras metodologías de diseño de filtros digitales resultan de migrar las técnicas de diseño de filtros analógicos al mundo digital, para las que, por provenir de un mundo concebido como continuo, su principal cambio consistió en la adición de métodos de discretización o conversión de sus modelos al dominio del tiempo discreto. Uno de los resultados fue la concepción por separado del proceso de cuantificación de los coeficientes (o parámetros) del filtro y el proceso de diseño en sí. Las teorías actuales, fundamentadas en las metodologías de diseño tradicionales, se refieren al diseño de filtros digitales como la determinación del conjunto de coeficientes que hacen parte de una ecuación en diferencias o de una función de transferencia que describe al filtro de interés, pero, suponen que dichos coeficientes son ideales, es decir, números reales, cuando en realidad, actualmente, con la diversificación y proliferación de las tecnologías digitales, las plataformas de implementación finales operan con números binarios, que por naturaleza son discretos y limitados en precisión decimal. Con el fin de remediar dicha situación, aparece la etapa de cuantificación al final del proceso de diseño, después de obtener los valores del conjunto de coeficientes del filtro. En síntesis, la cuantificación se encarga de ajustar los coeficientes reales a números binarios en la representación numérica y con la precisión binaria disponible en la plataforma de implementación final. Este es un proceso irreversible, pues es imposible conocer a qué número real exacto corresponde el valor cuantificado [1, 2].

Las pérdidas de información por cuantificación perturban la respuesta en frecuencia respecto a su valor original, comprometiendo la selectividad del filtro, a tal punto que puede no cumplir con las especificaciones originales del diseño, e incluso un filtro IIR puede llegar a ser inestable. En el caso IIR, que está representado por una función de transferencia racional, cada polo y cada cero queda determinado por todos los errores de cuantificación de los polinomios del numerador y del denominador, desplazándolos a otras posiciones del plano Z. Kaiser demostró (1966) que si los polos (ceros) están agrupados estrechamente (como ocurre con los filtros paso bajo o pasa banda de banda estrecha), es posible que pequeños errores en los coeficientes del denominador

(numerador) puedan causar grandes desplazamientos de los polos (ceros). Además, demostró que cuando mayor es el número de polos (ceros) agrupados, mayor es dicha sensibilidad [3]. La única propuesta que las metodologías tradicionales de diseño de filtros digitales FIR e IIR (y los *software* de diseño que en ellas se basan) ofrecen para mejorar la selectividad del filtro es aumentar su orden, pero, esto supone un consumo adicional de memoria, además de una mayor riesgo de inestabilidad del sistema implementado [1-5]. Desde el punto de vista del proceso de cuantificación, la solución consiste en aumentar la precisión en bits de los coeficientes, pero, esto también supone más memoria de almacenamiento [1-4]. Aumentar el espacio en memoria, probablemente implique mayores costos de implementación de los sistemas.

En esta tesis de investigación se proponen tres metodologías de diseño de filtros digitales lineales e invariantes en el tiempo, FIR e IIR, basadas en las herramientas de optimización metaheurística APBIL, GA y TS, orientadas a encontrar el diseño optimo, contemplando la etapa de cuantificación desde el comienzo y a lo largo del proceso de diseño, y no al final, como ocurres en otras metodologías, incluidas las metodologías tradicionales. Los resultados de las simulaciones son prometedores y muestran un buen desempeño de los filtros obtenidos, en términos del Error Medio Cuadrático (o MSE, por sus siglas en inglés) entre las respuestas en frecuencia deseada y la obtenida del filtro optimizado. En particular, las herramientas APBIL y GA demuestran tener la capacidad de proporcionar resultados que superan en calidad a las respuestas entregadas por las metodologías tradicionales, especialmente, cuando se trata de la implementación de filtros digitales con número reducido de coeficientes y/o baja precisión numérica.

De entre las metodologías propuestas, se destaca por su desempeño aquella que implementa el algoritmo de optimización APBIL. De entre los casos evaluados, en los cuales se igualaron las condiciones de tipo de sistema (FIR o IIR), el tipo de respuesta (paso bajo, paso alto, etc.), número de coeficientes, precisión y representación numérica, el algoritmo APBIL superó al diseño tradicional el 87.13% de las veces, el GA lo hizo el 77.14% y el algoritmo TS en un 55.4%. En relación a los tiempos de convergencia, el algoritmo APBIL fue el más veloz, pues no sobrepasó los 45 segundos, mientras que, el GA alcanzó el minuto y medio, y el algoritmo TS, en ocasiones, superó los 20 minutos.

Las pruebas de comparación también demostraron que existen casos en los cuales se pueden obtener mejores diseños, con un número menor de coeficientes, cuando se utilizan las metodologías propuestas (sobresaliendo el uso del algoritmo APBIL), que cuando se emplean las

metodologías tradicionales. La obtención de mejores diseños con un número menor de coeficientes (o memorias) implica que las metodologías de diseño propuestas (especialmente aquella que implementa el algoritmo APBIL), son capaces de reducir los costos de implementación de los sistemas requeridos, que llevado a un marco de producción en masa de tecnologías digitales de alto consumo, puede significar un ahorro económico significativo.

Como resultado de la implementación de los algoritmos APBIL, GA y TS para el diseño de filtros digitales LTI, se construyó un *software*, con una interfaz de usuario amigable, que permite realizar diseños FIR e IIR empleando cualquiera de las herramientas de optimización metaheurística propuestas. Dicho *software*, presenta las mismas ventajas que las metodologías de diseño presentadas, pues a diferencia de muchos otros *software* de diseño de filtros digitales, contempla la cuantificación y codificación desde un comienzo del proceso de diseño, y no solo al final del mismo. Adicionalmente, y de forma simultánea, permite obtener el diseño bajo las técnicas tradicionales más conocidas, proporcionando el valor de los coeficientes ideales (previos a la cuantificación) y cuantificados, que a su vez, utiliza para realizar comparaciones gráficas de los Espectros de Frecuencia y del MSE de los diseños obtenidos.

Este *software* de diseño, puede ser distribuido a diferentes profesionales, que estando interesados en obtener el diseño de un filtro digital para implementarlo en su campo respectivo, sin ser expertos en la temática, pueden obtener por sí mismos un diseño acorde a sus necesidades y con la mejor calidad posible. Gracias a que la interfaz de usuario desarrollada es de fácil manejo y contempla como valores por defecto los sugeridos por los resultados de los exhaustivos (y necesarios) procesos de sintonización efectuados en los algoritmos de optimización metaheurística propuestos.

El presente documento está organizado de la siguiente manera: El Capítulo 1, Estado del arte y marco teórico, hace una breve presentación de la historia que involucra la aparición y desarrollo de los filtros digitales hasta el día de hoy (ver las secciones 1.1 Historia (pasado, presente y futuro), 1.2 El diseño hoy, y 1.2.1 El problema de la cuantificación), e igualmente introduce el concepto de optimización metaheurística y menciona algunos casos en los que se han implementado este tipo de técnica en el diseño de filtros digitales en general (filtros adaptativos, LTI de coeficientes ideales, y cuantificados). El Capítulo 2, Montaje experimental, presenta la implementación de los algoritmos de optimización TS, GA y APBIL en el diseño de filtros digitales LTI (ver secciones 2.1, 2.2 y 2.3, respectivamente), también describe los procesos de sintonización a los cuales fueron

sometidos los algoritmos, con la finalidad de mejorar su propio desempeño (ver sección 2.4), y presenta el *software* de diseño desarrollado a partir de las metodologías propuestas (ver sección 2.5). En el Capítulo 3, se discute alrededor de los resultados de las pruebas de comparación y desempeño, de cuyas conclusiones ya se habló parcialmente. Finalmente, en el Capítulo 4 se reúnen las Conclusiones y recomendaciones, derivadas de la investigación.

1. Estado del arte y marco teórico

1.1 Historia (pasado, presente y futuro)

La historia de los filtros digitales está íntimamente relacionada con el desarrollo histórico del procesamiento de señales. El origen de los filtros digitales abre las puertas al procesamiento digital: en su momento, su creación incitó el interés creciente entre los profesionales de que las técnicas que, hasta entonces, solo pertenecían al diseño y procesamiento analógico, tuviesen su semejante en el procesamiento digital. En general, el procesamiento de señales debe su aparición a diversas técnicas matemáticas desarrolladas previamente (a lo largo de los siglos XVII y XVIII), y su crecimiento y evolución al desarrollo de la tecnología electrónica (análoga y digital), a la diversificación y proliferación de las tecnologías digitales en diversas aplicaciones y a su disponibilidad masiva, que incrementó sustancialmente el número de personas que trabajaban en el ámbito del Procesamiento Digital de Señales (PDS), a la demandante Segunda Guerra Mundial, y al desarrollo de las telecomunicaciones.

Existen muchos eventos y personajes relevantes en la historia de la humanidad, que también lo son en el marco del desarrollo histórico del procesamiento de señales. Al final del libro *Tratamiento Digital de la Señal Teoría y Aplicaciones* ([1]) se expone un breve, pero sustancial, resumen histórico del procesamiento de señales, abarcando hechos que van desde del siglo XVII hasta la década de los años 90. Algunos de esos hechos hacen mayor referencia al surgimiento de los elementos característicos de los filtros digitales, tales como: la concepción de las ecuaciones en diferencias, el análisis en frecuencia y los primeros desarrollos matemáticos considerados como los primeros filtros digitales de la historia. A continuación, se mencionan algunos, de manera cronológica, para contar un poco el cuándo o cómo surgen las bases fundamentales del diseño de filtros digitales:

Desde la invención del cálculo en el siglo XVII, se desarrollaron modelos basados en funciones continuas y ecuaciones en diferencias, que pretendieron explicar fenómenos físicos. En muchos

casos, la dificultad de encontrar una solución analítica impulsó la creación de *métodos numéricos* con el fin de hallar alguna solución. Por ejemplo: Newton utilizó ecuaciones en diferencias finitas, que son un caso particular de sistemas de tiempo discreto para resolver ciertos problemas. En el siglo XVIII, matemáticos como Euler, Bernoulli y Lagrange, desarrollaron métodos de integración e interpolación numéricos, actualmente considerados casos particulares de filtros digitales. En 1738, Bernouli resolvió la ecuación de la cuerda vibrante. En 1805, Gauss había descubierto el fundamento del algoritmo FFT (*Fast Fourier Transform*), el cual, Cooley y Tukey desarrollarían 105 años después de manera independiente; e incluso, Gauss logró descubrirlo antes de que el mismo Joseph Fourier publicara su tratado sobre la representación mediante series armónicas de funciones. Por otro lado, en 1822, Joseph Fourier presenta en su tesis titulada *Teoría Analítica del Calor* la extensión del problema de la ecuación de onda a funciones arbitrarias [1].

Casi al mismo tiempo, la observación de fenómenos repetitivos naturales, como el día y la noche, las estaciones climáticas, o las migraciones de aves, propiciaron la invención de calendarios, que reflejan la periodicidad de éstos eventos, de manera análoga al análisis espectral. El origen del concepto moderno de *espectro* se debe a los estudios de Pitágoras, sobre sus observaciones de la periodicidad de una cuerda vibrante, y a Newton, quién descubrió que la luz blanca se descompone en bandas de colores al pasar por un prisma, debido a las diferentes longitudes de onda (frecuencias) que se superponen en la luz blanca [1, 2]. Para el siglo XIX, comienza a desarrollarse diversas aplicaciones basadas en análisis armónico de fenómenos naturales cuasi periódicos como el clima, el sonido, el caudal de los ríos, las mareas o el número de manchas solares, entre otros. Junto a lo cual, se desarrollaron analizadores de armónicos mecánicos (1876), para la predicción. Un ejemplo de ello fue la construcción de un *analizador espectral*, en 1882, en Estados Unidos de Norte América, cuya labor fue elaborar tablas de mareas entre los años 1883 y 1910. En 1898, Schuster propone el *periodograma*, que constituye lo herramienta fundamental del análisis espectral denominado clásico, e identificó algunos efectos negativos de la técnica, tales como las altas varianzas y los lóbulos laterales debidos al enventanado [1].

Otros avances y hechos históricos impulsan aún más el desarrollo técnico y tecnológico del procesamiento de las señales: Después de la demostración de comunicación eléctrica y a distancia por F.B. Morse en el año 1844, y de la invención del teléfono, se inicia la construcción de las primeras redes de comunicación telefónica y telegráfica para cubrir grandes ciudades [1]. A la par, en 1864, el físico Inglés llamado James Clark Maxwell presenta su trabajo sobre electricidad y

magnetismo, con el que nacen las Radiocomunicaciones. Su trabajo postula teóricamente la existencia de ondas magnéticas, permitiendo explicar el carácter electromagnético de la luz, al igual que la propagación de la energía eléctrica en forma de onda por el espacio, lo cual, en 1888, Heinrich Hertz demostraría experimentalmente [1]. El final del siglo XIX está marcado fuertemente por la Segunda Guerra Mundial, que trajo consigo grandes avances en prácticamente todas las ramas de la técnica, desde aeronáutica hasta comunicaciones. Por ejemplo, las comunicaciones FM se generalizan, e igualmente se desarrollan el radar y el sonar, basados en la transmisión y recepción de pulsos, y en proporcionar información discreta. Después del fin de la guerra, en 1948, se inventa el transistor, Hamming inventa los códigos de corrección de errores y Shannon establece la capacidad de información que es posible transmitir por un canal sin errores [1].

En 1948, en Inglaterra se construye el primer prototipo de un ordenador electrónico con programa almacenado, convirtiéndose en el evento que abre la brecha para la aparición del procesamiento digital. Durante la década de los años 50, a pesar de que la escasez y las limitaciones de estos ordenadores impedían su uso en lo que propiamente hoy se conoce como PDS, se utilizaron para simular el comportamiento de sistemas analógicos antes de ser construidos. Permitiendo probar nuevos algoritmos y estrategias de procesado de forma flexible, sin comprometer costosos recursos. Un ejemplo de dichas simulaciones fueron los vocoders simulados en los laboratorios Lincoln o Bell [1].

La idea de simular el mundo analógico, creó una fijación entre ingenieros de lograr hacer de forma discreta lo mismo que se hacía de forma analógica. Por ejemplo, en la década de los años 50 se desarrollan los métodos de filtros digitales por Kaiser, Rabiner, Oppenheim y Gold, entre otros, con la finalidad de obtener el mismo potencial de diseño de filtros digitales que se tenía para los filtros analógicos. La exploración y explotación de este concepto, junto con la flexibilidad de los ordenadores, permitieron la aparición de nuevas técnicas de diseño, que no deben su origen al mundo analógico [1]. Adicionalmente, en la década de los 60, se desarrollan muchos otros algoritmos de procesamiento de señales, entre ellos: el descubrimiento de James Colley y Jhon Tukey de la FFT (1965), y el desarrollo de la transformada Chirp-Z (1968) [1]. Por otro lado, en la época de los 70 surgen las primeras obras literarias que constituyeron las primeras referencias para la mayoría de las universidades importantes: en 1975 se escribe el primer libro, *Digital Signal Processing* de Oppenheim y Schaffer, entre otros textos renombrados de la época como *Digital*

Processing of Speech Signals de Rabiner y Schaffer, y *Theory and Applications of Digital Signal Processing* de Rabiner y Gold [1].

Posteriormente, desde las década de los años 60, se han realizado grandes desarrollos tecnológicos, que van desde la puesta en órbita del primer satélite (el Telstar, en 1962, por la AT&T), pasando por los primeros videojuegos (1972) y juguetes electrónicos con procesamiento de voz (el *Speak & Spell*, en 1978, por Texas Instrument), hasta nuevos modelos de computadoras personales, dispuestas para el PDS, entre otros. En relación al progreso computacional, hasta fines de la década de los 80, se dieron los siguientes avances: En 1971 Intel produce el primer chip microprocesador (no apto para el PDS, porque sus procesos requerían varios ciclos de instrucción), en 1979 Intel presenta el monochip de PDS Intel 2920, con sendos convertidores A/D (conversión de señales analógicas a digitales) y D/A (conversión de señales digitales a analógicas) de 9 bits (pero, con un rango de aplicaciones limitado, debido a la falta de un multiplicador de *hardware* y la escasa resolución del interfaz analógico), en 1981 IBM saca al mercado el primer ordenador personal con sistema operativo DOS, y en 1983 se presenta el PC-XT, el primer ordenador personal con disco duro fijo. Además, las principales empresas fabricantes de conductores, Texas Instrumen, Motorola, NEC y ATT sacaron sus primeros monochips de PDS, lo cual favoreció el crecimiento de la cantidad de aplicaciones [1]. Entre el principio de la década de los 80 y 1990, el número de ordenadores personales pasó de 1.3 a 115 millones, e igualmente la industria del *software* se desarrolló fuertemente; la disponibilidad creciente de medios de cálculo económicos favoreció drásticamente el número de personas que trabajaban en el ámbito del PDS [1].

Poco a poco, el procesamiento de señales en general se fue fortificando, y con la aparición del procesamiento digital, se fue extendiendo a más áreas de conocimiento [2]. Las ventajas del procesamiento digital sobre el procesamiento analógico, tales como la inmunidad frente al ruido, estabilidad frente a temperatura, repetitividad, flexibilidad, etc., han favorecido su predilección para diferentes aplicaciones, e incluso, han ocasionado la migración de la mayoría de tecnologías de origen analógico al mundo digital [3]. Así, desde la década de los 90, se han sumado muchas invenciones y desarrollos técnicos y tecnológicos ligados al PDS, para una amplia gama de aplicaciones. Actualmente, existe una demanda creciente en *software* y *hardware* de mayor complejidad en el PDS para aplicaciones militares, gubernamentales, ingenieriles, industriales y en la electrónica de consumo, de bajo coste y altos volúmenes de venta [1-3]. A futuro, las nuevas tecnologías, y en especial los desarrollos computacionales, permitirán mejorar todo en cuanto al

procesamiento de señales se refiere, incluyendo los procesos de diseño de los mismos filtros digitales (el cual, es uno de los puntos que motiva la presente tesis). Una visión futurista, tal como la búsqueda de la *computadora cuántica*, promete ser un punto de quiebre revolucionario en el progreso de la humanidad y para el procesamiento de las señales, y al igual que en el pasado, como ocurrió entre los mundos analógico y digital, significará la evolución del concepto del filtro electrónico (que para entonces tal vez sea llamado como el *filtro cuántico*).

1.2 El diseño hoy

Retornando a la actualidad, y tomando como objeto de estudio los filtros digitales lineales e invariantes en el tiempo, su diseño consiste en la determinación de los parámetros o coeficientes de una función de transferencia o de una ecuación en diferencias que lo representa y que se aproxima a la respuesta en frecuencia deseada, dentro de unas tolerancias específicas. Según dicha respuesta los filtros digitales, y en general, cualquier sistema representado por ecuaciones en diferencias se puede clasificar en dos categorías: aquellos cuya Respuesta al Impulso es Finita (FIR) y aquellos cuya Respuesta al Impulso es Infinita (IIR) [1-4]. Las Ecuaciones (1-1) y (1-2) presentan las formas genéricas de las ecuaciones en diferencias en tiempo discreto de los filtros digitales IIR y FIR, respectivamente. Donde x_{n-i} representa las entradas del sistema, y_{n-j} las salidas que se realimentan, b_i y a_j los respectivos coeficientes, N el número de ceros y M el número de polos del sistema. El orden de la ecuación en diferencias queda definido por el máximo valor entre N y M.

$$y_n = \sum_{i=0}^{N} b_i x_{n-i} + \sum_{j=0}^{M} a_j y_{n-j} \qquad (1\text{-}1) \qquad\qquad y_n = \sum_{i=0}^{N} b_i x_{n-i} \qquad (1\text{-}2)$$

Al aplicar la transformada Z en las ecuaciones anteriores se obtén las Ecuaciones (1-3) y (1-4), referidas como las funciones de transferencia de los filtros digitales IIR y FIR, respectivamente.

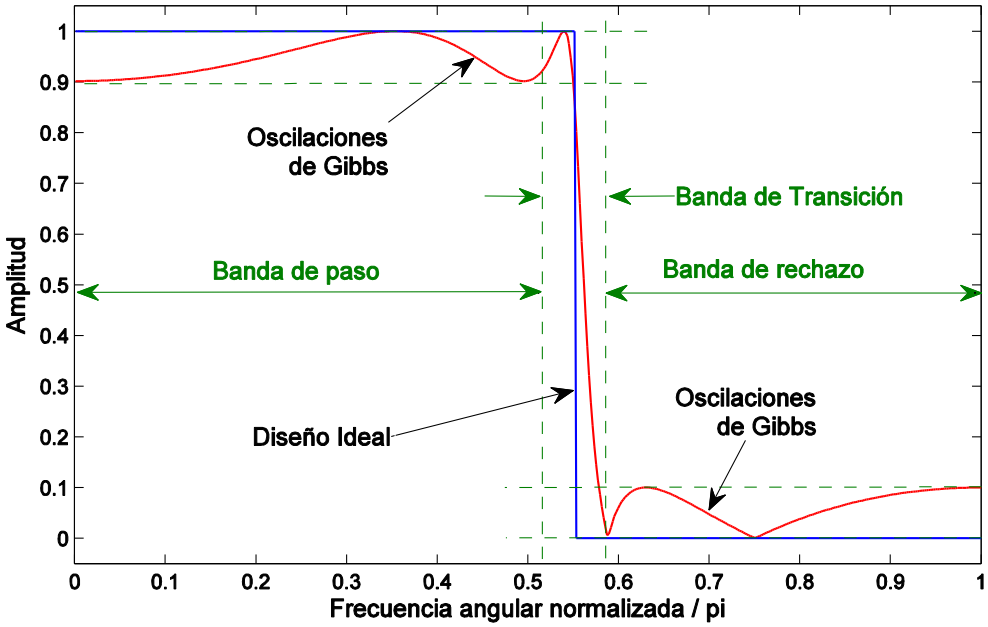
$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 - a_1 z^{-1} - \cdots - a_M z^{-M}} \qquad (1\text{-}3) \qquad H(z) = b_0 + b_1 z^{-1} + \cdots + b_N z^{-N} \qquad (1\text{-}4)$$

El orden de la ecuación en diferencias (o de la función de transferencia), también conocido como el orden del filtro, determina el mínimo número de bloques de memoria que el filtro diseñado demanda para su implementación [1-4]. Para una misma aplicación se pueden obtener diseños de filtros de tipo FIR o IIR. En comparación, para obtener el mismo rendimiento, un diseño basado en filtros tipo IIR requiere menos bloques de memoria, pero puede llegar a ser inestable para implementaciones en las que la precisión de los coeficientes es baja. Por otro lado, el diseño de

filtros FIR usualmente requiere de un mayor número de bloques de memoria, sin embargo, su ventaja radica en que siempre es estable y puede ser implementado con el uso de coeficientes de baja precisión [1-5].

Todos los diseños de filtros digitales pretenden aproximarse al diseño de un filtro ideal, cuya respuesta en frecuencia ostenta una forma rectangular. Teóricamente, la representación en el dominio del tiempo del filtro ideal requiere de una infinidad de parámetros o coeficientes; pero, los limitantes de la realidad se imponen, porque es imposible almacenar un número infinito de información, o dicho de otra manera, es imposible disponer de infinitos bloques de memoria. Dichos limites ocasionan la aparición de una banda de transición y oscilaciones en las bandas de paso y rechazo, referidas como oscilaciones de Gibbs [1-5] (ver Figura 1-1).

Figura 1-1: Respuesta ideal y oscilaciones de Gibbs en el dominio de la frecuencia.

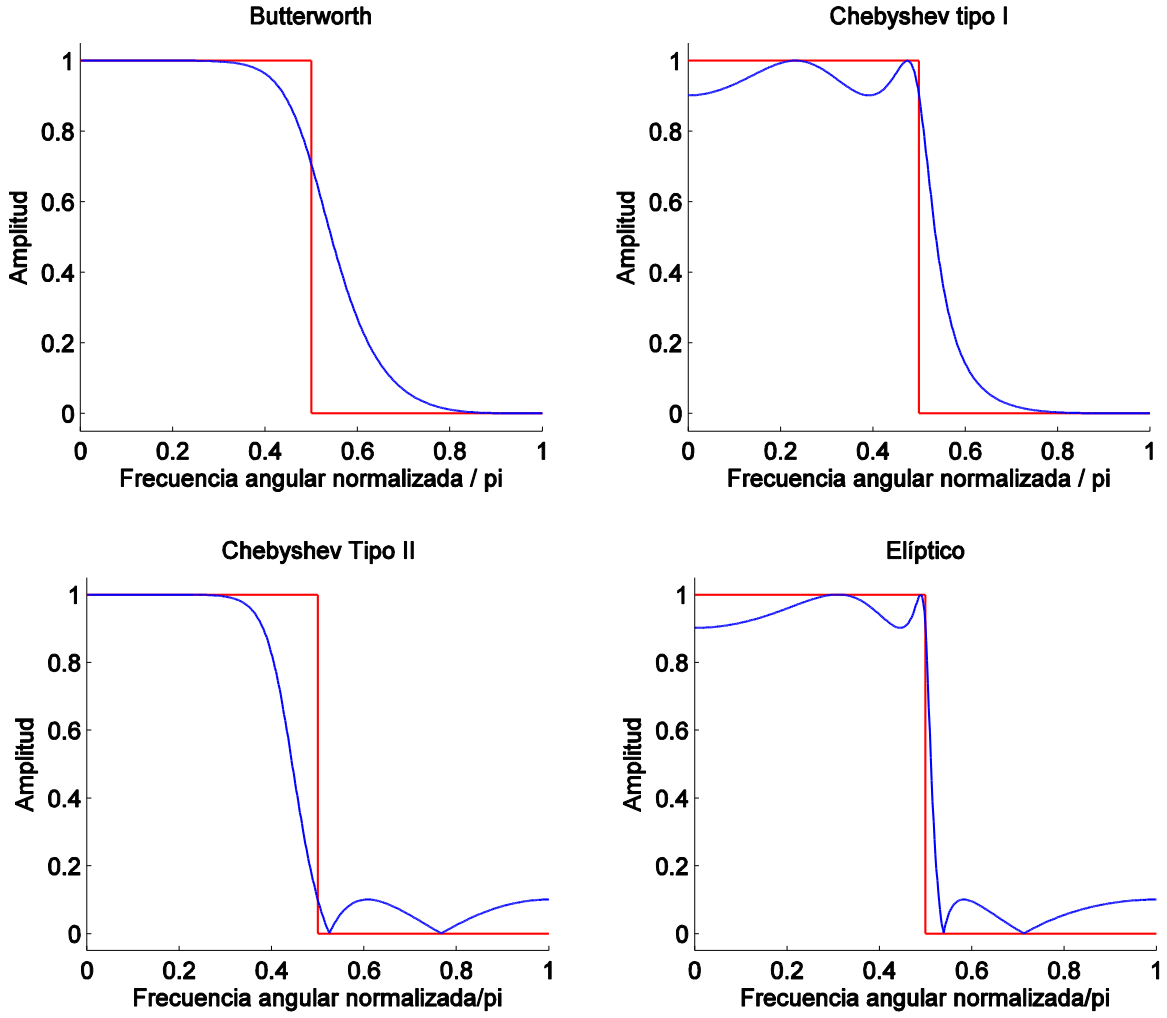


Nombre de la fuente: Adaptado de [1-4].

Producto de la migración del mundo analógico al digital, las metodologías tradicionales de diseño de filtros IIR digitales se basan en las metodologías de diseño de filtros IIR analógicos. De allí las formas más comunes para los filtros digitales: filtros Peak, Notch, Butterworth, Chebyshev tipo I y tipo II, Bessel y elípticos. Estos son distinguibles principalmente por sus diferencias en relación a

las oscilaciones en sus bandas de paso y rechazo, y el ancho de la banda de transición [1-4] (ver Figura 1-2).

Figura 1-2: Respuesta típica de los filtros IIR digitales más comunes.



Nombre de la fuente: Adaptado de [1-4].

Las metodologías tradicionales de diseño de filtros FIR se originaron en el mundo digital, y no tienen símil en el mundo analógico. Éstas se pueden clasificar en tres grandes grupos: los métodos de enventanado (de fase lineal), los métodos de muestreo en frecuencia y los métodos de rizado constante. Los métodos de enventanado aprovechan que la respuesta impulsiva de un filtro ideal (o IIR) se aproxima a cero conforme crecen hacia el infinito, para truncar su respuesta cuando el número de parámetros o coeficientes ya no brindan aportes significativos; en otras palabras, se enfocan en producir una aproximación de la respuesta temporal deseada (ideal o IIR) [1-5]. Estos usan el concepto de ventana para muestrear la respuesta deseada en el dominio del tiempo a partir

de la respuesta impulsiva. Existe una amplia colección de ventanas, propuestas por diversos autores, de los cuales se han ganado su nombre. Cada una de éstas, con sus debilidades y fortalezas, intenta equilibrar los efectos de las oscilaciones de Gibbs y de la pendiente que se produce en la banda de transición [1-4]. Por otra parte, los métodos de muestreo en frecuencia consisten, como su nombre lo indica, en tomar muestras en el dominio de la frecuencia de la respuesta ideal (o IIR), con el fin de que el diseño y la respuesta deseada coincidan de forma exacta en el conjunto de frecuencias muestreadas, y de alguna u otra forma en puntos intermedios; con el uso de la Transformada Discreta de Fourier (DFT) inversa, se obtiene el conjunto de coeficientes que caracterizan al filtro FIR [1-4]. Por último, se encuentran los métodos de rizado constante, que son un conjunto de algoritmos cuyo propósito es encontrar, para un orden determinado, el conjunto de coeficientes que mejor se aproxime a la respuesta deseada, sujeto a que la amplitud del rizado (u oscilaciones de Gibbs) se mantenga constante, con el fin de distribuir el error de aproximación de manera equitativa a lo largo del rizado, y así minimizarlo. Uno de los más mencionados es el algoritmo Parks-McClelan, también referido como el método de Remez [1-4].

Existen otras metodologías de diseño de filtros digitales FIR e IIR, basadas en el método de Mínimos cuadrados. En el caso del diseño FIR, Permite el control explícito de las frecuencias de los bordes banda y de las ganancias de cada banda para producir filtros de múltiples bandas, con bandas de transición arbitrarias. Internamente, su arquitectura es similar a la empleada por el algoritmo de Parcks-MacClelan, deduce fórmulas para estimar los coeficientes del filtro, sin embargo emplea el criterio de error cuadrático, mientras que en el otro se usa el criterio de error Chebyshev. Por otro lado, en el caso IIR, donde también es referido como diseño Yulewalk, permite el control explícito de los bordes de banda para producir filtros de múltiples bandas, e internamente, maneja la estructura original del método de mínimos cuadrados para arreglos vectoriales [1-4, 6].

Para mayor claridad sobre las metodologías de diseño mencionadas hasta ahora, se recomienda consultar las referencias de la [1] a la [4], o similares. En el presente trabajo no se ahondará más allá de lo que ya se ha citado (los objetivos que se plantean son diferentes). Todos los gráficos y cálculos presentados, que se refieran a dichas metodologías, fueron obtenidos a través de las rutinas disponibles en Matlab R2013a, para Windows. Para mayor información sobre rutinas y aplicaciones de Matlab para el PDS, incluido el diseño de filtros digitales, se recomienda consultar el libro *Digital Signal Procesing using Matlab* de J.G. Proakis y V.K. Ingle [6].

1.2.1 El problema de la cuantificación

Ya sea en una computadora personal o en alguna otra plataforma de implementación, tanto las señales como los coeficientes del filtro deben ser almacenados temporal o permanentemente, para lo cual se requiere representar el valor de la señal y de los coeficientes del filtro en números binarios. Las herramientas disponibles para el diseño de filtros digitales, computadoras personales y sus *software* (tales como Matlab), emplean una precisión arbitraria para la representación de los coeficientes, es decir, los consideran como números reales, sin contemplar que la plataforma de implementación final puede tener limitaciones para la representación numérica, ya sea de punto fijo o punto flotante [1-5].

Lo anterior ocurre como correspondencia a que todas las metodologías presentadas hasta ahora, las cuales conforman el ramillete de opciones de diseño de filtros digitales FIR e IIR aceptadas tradicionalmente, suponen una precisión infinita (números reales), en otras palabras, coeficientes ideales. Se emplea la cuantificación y codificación para trasladar dichos coeficientes al formato numérico utilizado por la plataforma de implementación final, cuyos limitantes en memoria presumen una pérdida de información que puede afectar el desempeño del filtro diseñado [1-5].

La cuantificación consiste en ajustar los valores de los coeficientes (o las muestras de una señal) a los niveles disponibles en la representación numérica elegida. Normalmente se realiza una cuantificación uniforme, empleando una representación de punto fijo, en donde todos los intervalos entre niveles tienen el mismo ancho [1-5]. Sin embargo, en el presente trabajo también se contempla la representación de punto flotante, que a diferencia de la representación de punto fijo, no posee una precisión uniforme: entre más cerca se esté del cero, mayor es la densidad de números representables, es decir, mejor es la precisión, y al alejarse hacia más o menos infinito se pierde precisión [6]. El proceso de cuantificación es irreversible, pues es imposible conocer a qué número real exacto corresponde el valor cuantificado, ya que no se conocen otros valores dentro de los intervalos [1].

Por otro lado, el proceso de codificación consiste en asignar a cada nivel de la cuantificación una palabra binaria. Éste es un proceso reversible y desde el punto de vista del PDS no importa la palabra binaria concreta que use para la representación [1]. En el caso de la cuantificación uniforme, se emplea una representación binaria de complemento a dos. Por otro lado, la representación de punto flotante estará basada en el estándar IEEE 754 de 1985 ([7]). Finalmente,

los bits producidos por la codificación son los que físicamente se almacenan, se manipulan por el *hardware* o se transmiten.

Las pérdidas de información por cuantificación perturban la repuesta en frecuencia respecto a su estado original, comprometiendo la selectividad en frecuencia, a tal punto que puede no cumplirse en absoluto las especificaciones originales del diseño, e incluso un filtro IIR puede llegar a ser inestable. En el caso IIR, que está representado por una función de transferencia racional, cada polo y cada cero queda afectado por todos los errores de cuantificación de los polinomios del numerador y denominador, desplazándolos a otras posiciones en el plano z. Kaiser demostró (1966) que si los polos (ceros) están agrupados estrechamente (como ocurre en los filtros pasa banda o paso bajo de banda estrecha), es posible que pequeños errores en los coeficientes del denominador (numerador) puedan causar grandes desplazamientos en los polos (ceros). Además, demostró que cuanto mayor es el número de polos (ceros) agrupados, mayor es la sensibilidad [3]. La única propuesta que las metodologías tradicionales de diseño de filtros digitales FIR e IIR (y los *software* de diseño que en ellas se basan) ofrecen para mejorar la selectividad del filtro es aumentar su orden, pero, esto supone un consumo adicional de memoria, además de un mayor riesgo de inestabilidad del sistema implementado [1-5]. Desde el punto de vista del proceso de cuantificación, la solución está en disminuir la distancia entre los niveles de cuantificación, aumentando la resolución en bits de los coeficientes, pero, esto también supone más memoria de almacenamiento [1-4]. Aumentar el espacio en memoria, probablemente implique mayores costos de implementación de los sistemas.

A raíz de los problemas que plantea la cuantificación al final del proceso de diseño, y siendo la única alternativa contemplada por las metodologías tradicionales, en el presente trabajo se propone una metodología de diseño de filtros digitales FIR e IIR basada en estrategias de optimización metaheurística, que contempla los procesos de cuantificación y codificación de los coeficientes del filtro desde el comienzo y a lo largo del proceso de diseño, y procura encontrar el arreglo de coeficientes óptimo para la respuesta en frecuencia deseada. Tanto para la cuantificación uniforme y no uniforme, basadas en las representaciones binarias de *punto fijo – complemento a dos* y *punto flotante – estándar IEEE 754.*

1.2.2 La optimización metaheurística

Los algoritmos de optimización metaheurística, también referidos como algoritmos evolutivos, son un conjunto de técnicas de búsqueda que mezclan estrategias matemáticas y heurísticas. Sus componentes de aleatoriedad impiden que exista una demostración formal del porqué funcionan, a diferencia de los métodos de optimización analítica (como el método de Mínimos Cuadrados y el algoritmo de Parcks-MacClean), que se sustentan en estrategias netamente matemáticas y por ende existe comprobación o prueba del porqué pueden alcanzar el óptimo. A pesar de ello, poco a poco toman mayor popularidad en la solución de problemas de búsqueda complejos, en ésta y otras áreas del conocimiento. Se prefieren cuando el conjunto de alternativas a solución o espacio de diseño es muy grande, ya que están en la capacidad de proveer soluciones iguales o próximas al óptimo en un tiempo prudente, inferior al invertido por otras estrategias de búsqueda. Existen problemas donde se ha demostrado que los métodos de búsqueda tradicionales pueden demorar semanas, meses e incluso varios años, resolviendo el mismo problema que un algoritmo de optimización metaheurística puede resolver en contados minutos o segundos; e incluso, aún es cierto si se recurre a la programación en paralelo (en la cual se dispone de varios núcleos de procesamiento independientes, que pueden realizar diferentes cálculos simultáneamente), como es el caso presentado en [8], para la reconfiguración de sistemas fotovoltaicos para la maximización de la energía producida.

La mayoría de algoritmos optimización metaheurística son bio-inspirados, sus estrategias resultaron de la observación detallada de sistemas de la naturaleza y la optimación de sus procesos innatos. De allí nombres como: Algoritmos Genéticos (GA, *Genetic Algorithms*), algoritmo de Optimización Colonia de Hormigas (ACO, *Ant Colony Optimisation*), algoritmo Colonia Artificial de Abejas (ABC, *Artificial Bee Colony*), Optimización de Enjambre de Partículas (PSO, *Perticle Swarm Optimization*) inspirado en el comportamiento de las bandadas de aves al volar, y el algoritmo de Recocido Simulado (SA, *Simulated Annealing*), también conocido como Enfriamiento Simulado, que se inspira en la reorganización de las partículas de un bloque de metal al enfriarse lentamente. Otros algoritmos de optimización metaheurística se inspiran en sistemas artificiales, o son resultado de la combinación de varios de estos algoritmos y técnicas de redes neuronales artificiales.

Cada algoritmo evolutivo usa su propia estrategia de búsqueda, diferente a las demás, así que se dispone de tantas como algoritmos evolutivos existen; pero, inicialmente se les puede catalogar en

poblacionales y de gradiente. Los algoritmos de gradiente, tales como el algoritmo SA y Búsqueda Tabú (TS, *Tabu Search*), emplean una única solución por cada iteración para realizar la búsqueda. Dicha solución se "desplaza" por el espacio de soluciones, según un delta o gradiente, que es calculado en una o en múltiples direcciones para determinar cuál es el siguiente paso. Cuando el objetivo es la minimización (maximización), sus estrategias de búsqueda les permite salir de aquellos puntos referidos como "mínimos locales" ("máximos locales"), atributo que no poseen las estrategias de optimización analíticas, que en su mayoría también son de gradiente. Por otro lado, los algoritmos evolutivos poblacionales, tales como PSO y ACO, emplean varias soluciones por iteración para realizar la búsqueda; normalmente dichas soluciones son aleatoriamente distribuidas por el espacio de soluciones para así abarcar una mayor área de búsqueda de manera simultánea. Conforme se desarrolla la búsqueda, se descartan áreas donde no hay progreso y se intensifica en zonas prometedoras hasta encontrar el óptimo. A diferencia de las técnicas de gradiente, la estrategia de los algoritmos poblacionales les permite ser más efectivas y eficientes, pues el riesgo de estancarse en un "óptimo local" es menor y el tiempo se reduce considerablemente.

Cada algoritmo tiene asociadas estrategias de exploración y explotación, la efectividad de éstas en cada uno empieza a marcar la diferencia. La exploración es la capacidad que el algoritmo posee de inspeccionar un número mayor de áreas del espacio de soluciones; algo que es vital en la etapa temprana de la búsqueda, para identificar zonas potenciales. Por su parte, la explotación es la capacidad de promover una (o un conjunto) solución, considerada la mejor opción, para así reducir los tiempos de computo o convergencia. La primera estrategia tiene como finalidad aumentar la probabilidad de encontrar una buena solución, en otras palabras, le apunta a la calidad, mientras que la segunda propende por la rapidez (ambos criterios son opuestos).

Haciendo una revisión bibliográfica en el PDS, se pueden encontrar diferentes aplicaciones en donde se emplea este tipo de técnica. Para el caso específico del diseño de filtros digitales, se expone su implementación en el diseños de filtros adaptativos (los cuales son filtros cuyos coeficientes varían con el tiempo), [9-10], al igual que en el diseño de filtros lineales e invariantes en el tiempo, sin considerar la cuantificación de los coeficientes, [11-16], y en menor porcentaje, el diseño de estos últimos (que son el objeto de estudio en la presente tesis), considerando la cuantificación, [17, 18]. Por ejemplo, en [9] y [10] se presenta la implementación de los algoritmos PSO y TS, respectivamente, para el diseño de filtros IIR adaptativos. En relación a los filtros digitales lineales e invariantes en el tiempo, en las referencias de la [11] a la [15] se presenta el

diseño de filtros IIR sin cuantificación, es decir, de coeficientes reales, empleando los algoritmos ABC [11], Búsqueda Gravitacional (GSA, *Gravitional Search Algorithm*) [12], Imperialista Competitivo (ICA, *Imperialist Competitive Algorithm*) [13], Inmunidad Artificial (IA, *Artificial Immune*) [14], inspirado en la resistencia de los individuos o especies a microorganismos patógenos, y el Hibrido Taguchi – Algoritmo Genético (HTGA, *Hybrid Tabuchi Genetic Algorithm*) [15], que es una mezcla de los algoritmos tradicionales Taguchi y Genético. Un número menor de trabajos son dedicados a los diseños FIR, como en la referencia [16], que presenta el diseño de filtros FIR, sin cuantificar, implementando los algoritmos PSO y de Evolución Diferencial (DE, *Differential Evolution*). Por otro lado, las referencias [17] y [5] presentan diseños de filtros digitales lineales e invariantes en el tiempo, que contemplan la cuantificación desde el comienzo, la primera se refiera al diseño de filtros IIR bajo el algoritmo ACO, y la segunda al diseño de filtros FIR bajo el algoritmo de Aprendizaje Incremental Basado en Poblaciones (PBIL, *Population-Based Incremental Learning*), inspirado en técnicas de inteligencia artificial, GA y en Algoritmos de Estimación de Distribuciones (EDA, *Estimation of Distribution Algorithms*). En ambos casos emplea la representación binaria de punto fijo para la cuantificación. En la mayoría de los trabajos mencionados, se ve la comparación de los resultados obtenidos bajo el algoritmo de interés respecto a otros algoritmos del éste tipo u otras versiones modificadas de él mismo, y es común encontrar comparaciones con relación a AG, PSO, SA y TS, por ser de los primeras y más renombradas técnicas de optimización metaheurística.

En el presente trabajo se expondrá la implementación de las herramientas de optimización PBIL adaptativo (APBIL), GA y TS como metodologías de diseño de filtros digitales lineales e invariantes en el tiempo FIR e IIR. A raíz de una revisión bibliográfica de las técnicas de optimización metaheurística existentes se eligen las herramientas TS y GA por ser referentes, y emplear estrategias de exploración destacadas, entre otras características ventajosas que más adelante se explicaran con mayor detalle. Ambas técnicas, que ya han sido empleadas en el diseño de filtros digitales, servirán en la comparación de rendimiento del algoritmo APBIL, que hasta ahora, no ha sido empleado para dicho fin. Vale la pena aclarar que, en la referencia [5] se expone el uso del algoritmo PBIL clásico o tradicional (TPBIL), y de manera exclusiva para diseños FIR. Adicionalmente, en la misma y otras referencias en las que se implementó la versión clásica del algoritmo, se sugiere, en la sección de trabajos futuros, la implementación de la versión adaptativa para mejorar el tiempo de convergencia, y además, como queda demostrado en el capítulo siguiente, también mejora sustancialmente la calidad de las soluciones entregadas. En general, el algoritmo PBIL se destaca en la resolución de problemas complejos en diferentes contextos: en el

diseño de controladores de Sistemas Eléctricos de Potencia [18], partición dinámica de *Hardware/Software* [29], mapeo y programación de tareas en redes al interior del circuito integrado (NoC, *Network on Chip*) [20], por mencionar solo algunos ejemplos. Adicionalmente, su estructura también ofrece ventajas que lo destacan por sobre muchos otras algoritmos de optimización metaheurística, como la manera particular de representar la población, sus estrategias de exploración y explotación, su número reducido de parámetros, entre otros.

En el siguiente capítulo (Montaje experimental) se presentan los elementos principales de las herramientas de optimización elegidas (TS, GA y APBIL), su implementación en el diseño de filtros digitales FIR e IIR y el desarrollo de una aplicativo basado en dichas metodologías de diseño.

2. Montaje experimental

A continuación, se presentan tres metodologías de diseño de filtros digitales lineales e invariantes en el tiempo FIR e IIR, basadas en las herramientas de optimización metaheurística APBIL, GA y TS. Las cuales consideran los recursos de *hardware* disponibles y están en la capacidad de brindar un diseño cercano (o igual) al óptimo. Su ventaja principal, en contraste con las metodologías de diseño tradicionales, radica en que el proceso de cuantificación es considerado a lo largo del proceso de diseño, y no al final. La estrategia general consiste en una búsqueda sistemática, dentro del espacio de diseño, de la solución que mejor cumpla el criterio de selección, que en este caso se refiere al mínimo error respecto a la respuesta en frecuencia deseada.

El algoritmo TS es una técnica de optimización de búsqueda local (o por gradiente), que utiliza una estrategia basada en el uso de estructuras de memoria para escapar de los "óptimos locales", en los que se puede caer al "moverse" por el espacio de soluciones. Dirige la búsqueda basándose en la historia de ésta: Evita regresar a soluciones ya visitadas, aunque estás sean mejores que la solución de la iteración actual. Bajo su enfoque, un movimiento puede ser de mejor calidad si, por ejemplo, su frecuencia de ocurrencia en el pasado es baja o no ha ocurrido antes, permitiendo explorar nuevas regiones. Los principales criterios para finalizar el proceso de búsqueda (también referidos como criterios de parada) consisten en si el número de soluciones a visitar es reducido o nulo, o si por un espacio de tiempo prudente la mejor solución de cada iteración no es de una calidad superior a las ya registradas. La solución considerada el punto óptimo será la mejor solución encontrada a lo largo del proceso de búsqueda [10].

Por otro lado, los algoritmos genéticos son técnicas de optimización de búsqueda poblacional, inspirados en los procesos de reproducción de las especies a nivel cromosómico, al igual que en la adaptabilidad y supervivencia de los seres más aptos y la mortandad de los más débiles. Las herramientas características de esta técnica son sus operaciones de cruce y mutación, a través de las cuales se combina eficazmente la información de los individuos más aptos de la población y se

exploran nuevas regiones del espacio de soluciones. Iterativamente, las soluciones de mejor calidad producen soluciones nuevas (denominadas hijos) que sustituyen dentro de la población a las soluciones peor valoradas. Con lo cual la búsqueda cada vez es más selectiva, conduciendo a toda la población hacia la región con mayor potencial de que el óptimo se encuentre allí. Finalmente, se espera que la mejor solución de la población en la última iteración sea igual al óptimo del especio de soluciones. En éste punto, normalmente ocurre que no solo una solución es la mejor, varias de las soluciones de la última iteración son idénticas entre sí e iguales a la ella. A partir de éste hecho se define uno de sus principales criterios de parada, que se refiere a la baja diversidad de la población, lo que significa que las soluciones se han cerrado alrededor de un mismo punto, el cual deberá coincidir con el punto óptimo [15, 21, 22].

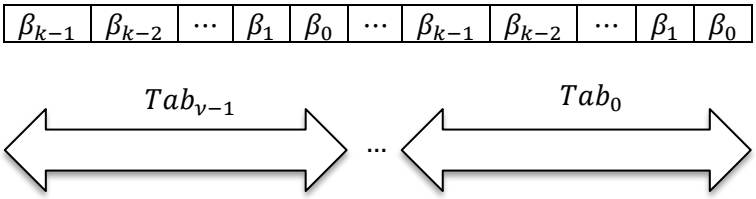
Por su parte, el algoritmo PBIL es una herramienta de búsqueda poblacional, que emplea las probabilidades de ocurrencia de las características de las mejores soluciones de cada iteración para guiar la búsqueda hacia el óptimo. Su principal diferencia respecto a otras técnicas poblaciones es su forma de representar la población: Unifica en un único arreglo matricial todo el espacio de soluciones, en el cual se almacenan las probabilidades de ocurrencia de cada solución posible. En síntesis, su dinámica consiste en que a partir del arreglo de probabilidades, en cada iteración se genera una nueva población de soluciones individuales, que luego son valoradas para tomar de ellas la mejor de todas y utilizarla para la actualización de las mismas probabilidades. De ésta manera, se aumentan las probabilidades de ocurrencia (en cada iteración) de las características que hacen parte de las mejores soluciones y al mismo tiempo, se reducen las probabilidades de ocurrencia de las características que no hacen parte de ellas. Conforme se actualizan las probabilidades, se cierra cada vez más la búsqueda sobre alguna zona potencial del espacio de soluciones. Hacia el final, algunas probabilidades tienden a la unidad y otras a cero, reflejando sobre el mismo arreglo de probabilidades la solución óptima [5, 18-20, 23].

La tasa a la que se actualizan las probabilidades es referida como Tasa de Aprendizaje (LR, *Learning Rate*), cuando dicha tasa es dinámica el algoritmo PBIL es referido como PBIL adaptativo. El algoritmo PBIL tradicional emplea una tasa constante. La versión adaptativa emplea una tasa de aprendizaje variable, convenientemente ajustada a la cambiante distribución de las probabilidades dentro del arreglo mismo [18, 24].

A pesar de su efectividad, ninguna de las metodologías de diseño de filtros digitales basadas en las herramientas de optimización metaheurística propuestas puede garantizar el hallazgo del óptimo absoluto. Por tanto, para reducir las posibilidades de fallo se ha realizado un proceso de sintonización sobre cada algoritmo, que brinda una mayor confianza de hallar buenas soluciones, próximas al óptimo (ver sección 2.4 Proceso de sintonización).

En primera instancia, hay que definir la representación de las soluciones que manipularan los algoritmos de optimización metaheurística, que a su vez representan los filtros digitales en proceso de diseño. Esta representación es el puente entre la herramienta metaheurística y el concepto de diseño de filtros digitales FIR e IIR. Dada la intención de controlar la cuantificación desde el principio, la representación de las soluciones debe considerar el formato numérico exigido por la plataforma de implementación final. Como se mencionó en el capítulo anterior (sección 1.2.1 El problema de la cuantificación), ésta corresponde a una representación numérica binaria. Así, cada solución será una cadena de dígitos binarios correspondientes a todos los coeficientes (o *tabs*) del filtro digital, ordenados en serie. En la Figura 2-1 se muestra la representación de la solución para el problema de diseño de filtros digitales FIR e IIR. Su tamaño queda determinado por el producto entre el número de coeficientes o *tabs* (ν) que constituyen el filtro y la resolución en bits (k) usada para representar cada *tab*.

Figura 2-1: Representación de la solución.



Nombre de la fuente: Elaboración propia

Para los sistemas IIR, los coeficientes quedan ordenados consecutivamente, primero el numerador y luego el denominador, comenzando de izquierda a derecha con b_0, hasta llegar a a_M, y el número total de *tabs* (ν) queda determinado por la suma de los órdenes del numerador y denominador más dos (ver Ecuación (2-1)). El caso FIR sólo involucra a los coeficientes b_i, por lo cual el total de *tabs* es igual al orden del filtro más uno (ver Ecuación (2-1))

$$\nu = \begin{cases} N + M + 2, & si\ es\ IIR \\ N + 1, & si\ es\ FIR \end{cases} \tag{2-1}$$

2.1 Algoritmos de Búsqueda Tabú (TS)

La estructura básica del algoritmo TS implementado se muestra en el Esquema Algoritmo 1 (ver Figura 2-2). La búsqueda se inicia con una solución semilla elegida de manera aleatoria. A partir de dicha solución se generan algunos o todos los vecinos numéricos (siempre y cuando no sean soluciones vetadas por pertenecer a la Lista Tabú o a la Memoria Atributiva). Se valora el conjunto de soluciones resultantes, referido como vecindario, y la mejor solución, referida como el mejor vecino, remplaza a la solución semilla, sin importar si es o no una mejor solución al problema de diseño. Las soluciones generadas son agregadas a la Lista Tabú (TL, *Tabu List*) y el movimiento realizado para obtener el mejor vecino es registrado en la Memoria Atributiva (AM, *Attributive Memory*). Mientras ningún criterio de parada se cumpla, se repite el ciclo, retornando al segundo paso, la creación de un nuevo vecindario a partir del mejor vecino.

Figura 2-2: Esquema general del algoritmo TS.

Algoritmo 1: Algoritmo TS.

Entrada: Una solución semilla $v \times k$ seed solution.
Salida: Un diseño potimo de un filtro digital IIR o FIR.
Inicio

$Neighbor = Initialize\_Seed(Rand)$;
$TL = Initialize\_Tabu\_List(Neighbor)$;
$AM = Initialize\_Attributive\_Memory(Void)$;

Repetir

$NBHD = Create\_Neigborhood(Neighbor, AM)$;
$Fitness = Evaluate\_Neigborhood(NBHD)$;
$Neighbor = Choose\_Best\_Neighbor(NBHD, Fitness, TL)$;
$TL = Update\_Tabu\_List(TL, NBHD)$;
$AM = Update\_Attributive\_Memory(AM, Neighbor)$;

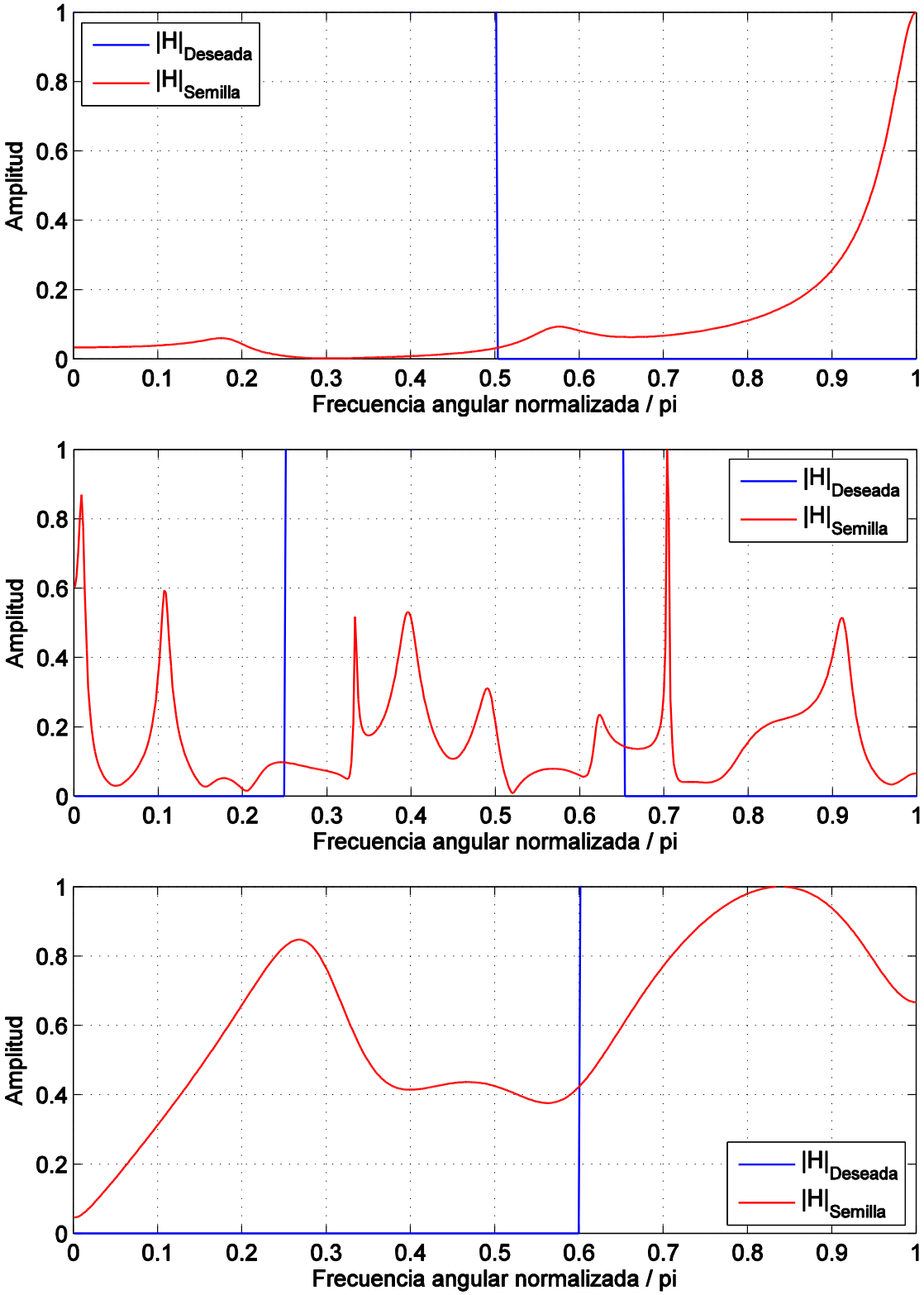
Hasta $(Stopping\_Criterion = false)$
$Best = Choose\_Best(TL)$;
Retornar $Best$;

Fin

Nombre de la fuente: Elaboración propia

La rutina ***Initialize_Seed*** genera una solución aleatoria de tamaño $v \times k$. Similar a la mostrada en la Figura 2-1. Las probabilidades de que la primera solución sea similar al filtro deseado son casi nulas, y normalmente es lo suficientemente errática para ni siquiera mostrar selectividad en frecuencia (ver Figura 2-3).

Figura 2-3: Respuesta en frecuencia de la solución inicial.



Nombre de la fuente: Elaboración propia

La rutina ***Create_Neigborhood*** genera el vecindario a partir de la solución referida como mejor vecino. Se produce realizando copias de la solución original y cambiando de valor un bit, uno diferente por cada copia. Los únicos bits que no podrán ser modificados serán los que prohíba la AM. En un comienzo, cuando no hay restricción, el tamaño del vecindario es tan grande como bits tiene la solución ($v \times k$ bits), y gradualmente se irá reduciendo, conforme actúa la AM, hasta alcanzar su tamaño mínimo ($minNBHDsize$).

El algoritmo TS implementado usa dos mecanismos de memoria: Explicita y de Atributos. Las rutinas ***Initialize_Tabu_List*** y ***Initialize_Attributive_Memory*** tienen la tarea de inicializar dichas memorias.

La Lista Tabú hace la función de Memoria Explicita (EM, *Explicit Memory*); lo cual quiere decir que, en ella se almacenan las soluciones de manera completa. Las soluciones registradas son las visitadas durante la búsqueda. Puede ser de dos tipos: de corto y largo plazo. La memoria de corto plazo permite que después de cierto tiempo las soluciones marcadas como prohibidas sean liberadas y nuevamente puedan ser elegidas para generar nuevos vecindarios. Esto implica limitar el tamaño de la TL y que cuando ésta alcance su tope, las soluciones más antiguas salgan para que las nuevas entren, al estilo de una lista o cola ordenada. Es así como el tamaño de la TL incide en el tiempo o número de iteraciones que un elemento permanece en ella. Por otro lado, la memoria de largo plazo retiene por un mayor tiempo o permanentemente las soluciones dentro de la TL. Lo cual se consigue con un tamaño grande de la TL o al no limitar su crecimiento. La memoria de largo plazo tiene dos estrategias asociadas: Intensificar y diversificar la búsqueda.

La Memoria Atributiva registra información acerca de los atributos o características que cambian al moverse de una solución a otra. En ella se registran las características de soluciones pasadas, con el objeto de dirigir la búsqueda y prohibir movimientos que permitan retornar a ellas. De esta manera no solo una solución es marcada como tabú; sino que, todas las soluciones que posean el mismo atributo o que puedan ser conseguidas con el mismo movimiento son prohibidas. En éste caso, identifica qué bit, al ser modificado, produjo al nuevo mejor vecino, y anula la posibilidad de que su valor vuelva a ser cambiado durante un número determinado de iteraciones. El número máximo de bloqueos ($maxlocks$) o iteraciones consecutivas en que se prohíbe la modificación del bit, es equivalente al número máximo de bits que pueden permanecer bloqueados al mismo tiempo,

por lo cual su número debe ser inferior, y a lo sumo igual, al total te bits menos dos (ver Ecuación (2-2)).

$$\mathbf{max}\{maxlocks\} = v \times k - 2 \quad (2\text{-}2)$$

De esta manera, el tamaño del vecindario se reduce a una tasa de una solución por iteración, hasta alcanzar su tamaño mínimo ($minNBHDsize$). Si se elige el límite para el número máximo de bloqueos (descrito por la Ecuación (2-2)), el tamaño mínimo del vecindario será de solo dos soluciones.

Cuando el vecindario alcanza su tamaño mínimo (cualquiera sea éste) existen dos alternativas: continuar la búsqueda, ahora con un tamaño de vecindario constante, o finalizarla en este punto. Este es uno de los criterios de parada normalmente contemplados para el algoritmo TS (en adelante referido como *criterio del tamaño mínimo*), debido a que para este momento ya debió revisar un número sustancial de soluciones diferentes.

Debido a que el tamaño de la representación o total de bis ($v \times k$ bits) cambia de un problema de diseño a otro, el número máximo de bloqueos será definido por medio de una fracción o porcentaje del total de bits que pueden ser bloqueados. Así, si $frlocks$ es la fracción de bits que se desea bloquear (definida entre 0 y 1), el número máximo de bloqueos queda determinado por la Ecuación (2-4).

$$maxlocks = \mathbf{round}\{frlocks \times (v \times k - 2)\} \quad (2\text{-}3)$$

Con la rutina ***Evaluate_Neigborhood*** se evalúa la aptitud de las soluciones del vecindario para la resolución del problema de diseño de diseño de filtros digitales planteado. En los tres algoritmos (TS, GA y APBIL), se cumple que dentro de esta rutina las soluciones pasan por los siguientes procesos, en el orden mencionado: Conversión decimal de los coeficientes, transformación al dominio de la frecuencia, normalización de la amplitud del espectro de frecuencia y cálculo del Error Cuadrático Medio (MSE, *Mean Squared Error*) respecto a la respuesta en frecuencia deseada (cuyo espectro también está normalizado y posee el mismo número de muestras). El MSE entre las respuestas en frecuencia del filtro calculado ($H_{Calculated}$) y la respuesta deseada ($H_{Desired}$) obedece a la Ecuación (2-4), en la cual n equivale al número de muestras de frecuencia (homogéneamente distribuidas) de cada respuesta.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\left|H_{Desired_i} - H_{Calculated_i}\right|\right)^2 \quad (2\text{-}4)$$

Con la rutina ***Choose_Best_Neighbor*** se selecciona al mejor vecino según el *fitness*, siempre y cuando no se encuentre dentro de la TL. Si el algún momento todo el vecindario producido se encuentra dentro de la TL la búsqueda finaliza.

Por último, se actualizan las memorias Explicita y de Atributos. La rutina ***Update_Tabu_List*** actualiza la TL y la rutina ***Update_Attributive_Memory*** actualiza la AM con las características o atributos que permitieron obtener el mejor vecino a partir del anterior.

Además de los criterios de parada mencionados anteriormente, se han contemplado un límite de iteraciones y un límite de tiempo (criterios de parada también incluidos dentro de los algoritmos restantes). Cualquiera que sea el criterio de parada, cuando alguno de ellos se cumpla, se finaliza la búsqueda y se selecciona a la mejor solución registrada dentro de la TL. Cuando adicionalmente se emplea un criterio elitista, la mejor solución encontrada es almacenada y actualizada en cada iteración tras evaluar el nuevo vecindario; así, sin importar si el tamaño de la TL es pequeño, no existe riesgo de olvidar a la mejor solución encontrada a lo largo del proceso de búsqueda. Cualquiera sea el caso, la solución elegida es considerada el óptimo y en este caso la solución al problema de diseño de filtros digitales.

La principal estrategia del algoritmo TS es el hacer uso de estructuras de memoria, que le permiten evaluar diferentes soluciones cada vez. A diferencia de la mayoría de las herramientas de optimización metaheurística poblacionales (incluidos el GA y el algoritmo APBIL), las cuales carecen de estrategias similares, llevándolos a evaluar innecesariamente varias veces las mismas soluciones (especialmente, hacia el final del proceso de búsqueda).

2.2 Algoritmo Genético (GA)

La estructura básica del GA implementado se muestra en el Esquema Algoritmo 2 (ver Figura 2-4). Se inicia con la generación de una población de soluciones aleatorias (cada solución tiene la forma mostrada en la Figura 2-1). Al ser valoradas, las soluciones son ordenadas según su aptitud. Una parte de la población (normalmente se elige la mitad), en donde se encuentran las mejores, son

consideradas las soluciones más aptas (o soluciones "buenas"). De entre ellas se seleccionan las que harán el papel de "padres" para la producción de nuevas soluciones. Las parejas de padres producen parejas de hijos y se genera la cantidad de hijos suficiente para sustituir los demás miembros de la población, considerados los menos aptos (o soluciones "malas"). Posteriormente, se mutan algunos de los genes de la población (que en términos de la representación binaria significa que, bajo una selección aleatoria, el valor de algunos bits de la población es modificado). Finalmente, si algún criterio de parada se cumple, termina la búsqueda y se entrega la mejor solución de la última iteración. De lo contrario, se retorna al segundo paso, en el cual se realiza la valoración de las soluciones para reorganizar la población.

La rutina $\boldsymbol{Initilize\_Population}$ genera una población de tamaño definido ($popsize$), de soluciones aleatorias. Mediante la rutina $\boldsymbol{Evaluete\_Population}$ se valora la aptitud de cada solución en el problema de diseño de filtros digitales planteado (tal como se explicó en la sección anterior). A continuación la población es ordenada según el *fitness* de cada solución mediante la rutina $\boldsymbol{Sort\_Population}$.

Figura 2-4: Esquema general del Algoritmo Genético.

Algoritmo 2: GA.

Entrada: Una población de soluciones aleatorias, llamada Pop.
Salida: Un diseño óptimo de un filtro digital IIR o FIR
Inicio
    $Pop = Initialize\_Population(Rand)$;

    Repetir
        $Fitness = Evaluate\_Population(Pop)$;
        $Pop = Sort\_Population(Pop, Fitness)$;
        $Parents = Select\_Parents(Pop, Fitness)$;
        $Offsprings = Mating(Parenst)$;
        $Pop = Update\_Population(Pop, Offsprings, Fitness)$;
        $Pop = Mutate(Pop)$;
    Hasta $(Stopping\_Criterion = false)$
    $Best = Choose\_Best(Pop)$;
    Retornar $Best$;
Fin

Nombre de la fuente: Elaboración propia

La rutina $\boldsymbol{Select\_Parents}$ selecciona las parejas de padres de entre la parte de la población considerada de soluciones "buenas". Los criterios para la elección pueden ser variados. En el GA propuesto se consideraron dos criterios: selección aleatoria y selección según el *fitness*. El primero elige al azar las parejas de padres. El segundo criterio otorga una calificación o peso a las

soluciones de manera decreciente según su *fitness*, aumentando la probabilidad de las soluciones mejor valoradas de ser elegidas como padres en varias ocasiones.

La rutina ***Mating*** se encarga del cruce genético y su finalidad es refinar la búsqueda; en otras palabras, su propósito consiste en explotar la información recolectada por los padres. Una vez los padres son reunidos en parejas, una copia de cada padre es dividida en partes más pequeñas y de manera intercalada las partes de un padre son intercambiadas con las partes de su pareja, para formar dos nuevas soluciones referidas como hijos (ver Figura 2-5). Las divisiones se realizan en los puntos de cruce, cuyas ubicaciones a lo largo de la solución se define de manera aleatoria en cada operación de cruce. El número de puntos de cruce ($numcp$) es fijo y se elige con antelación. Varios puntos de cruce permiten una mayor exploración del espacio de soluciones, a diferencia de uno solo, bajo el cual la exploración es menos intensa; pero, un único punto conserva y transfiere de mejor manera las características que identifican a los padres (considerados las mejores soluciones de la población). Así, y como se mostrará en la sección 2.4.2 Sintonización del algoritmo TS, la elección adecuada del número de puntos de cruce dependerá del tamaño de la solución ($v \times k$ bits). Posterior a la operación de cruce, la rutina ***Update_Population*** remplaza los miembros de la población considerados menos aptos con los hijos recién generados.

Figura 2-5: Ejemplo de la operación de cruce del GA.

$$\bullet\ Padres \rightarrow \begin{cases} x_{Mother} = [100111 \bullet 0101000111 \bullet 01 \cdots 011] \\ x_{Father} = [\mathbf{011101 \bullet 1101110011 \bullet 11 \cdots 000}] \end{cases}$$

$$\bullet\ Hijos \rightarrow \begin{cases} x_{Offspring1} = [100111 \bullet \mathbf{1101110011} \bullet 01 \cdots 011] \\ x_{Offspring2} = [\mathbf{011101} \bullet 0101000111 \bullet \mathbf{11 \cdots 000}] \end{cases}$$

Nombre de la fuente: Adaptado de [21]

La rutina ***Mutate*** modifica el valor de algunos bits la población. En el GA propuesto se dispone de dos opciones para ejecutar ésta operación: El método tradicional, que consiste en recorrer todas las soluciones y bits de la solución, para determinar de manera aleatoria si dicho bit sufrirá o no una modificación. En éste caso se determina un porcentaje o fracción de mutación ($frmut$) igual para todos los bits, se toma un número al azar por bit y si dicho número es inferior a la fracción (o porcentaje) el valor del bit en cuestión es modificado. La alternativa al método tradicional (en

adelante referida como método de *número fijo de mutaciones*) consiste en elegir aleatoriamente solo algunos bits para ser modificados. En este caso la fracción de mutación (o porcentaje) representa el número de bits que se modificaran en cada iteración. El número de bits será fijo, más la elección de las soluciones y bits a modificar es aleatoria. Pruebas realizadas durante la construcción del algoritmo revelaron que el promedio del número de mutaciones proporcionadas por ambos métodos es equiparable y sus efectos sobre la calidad de la solución no son distinguibles. Pero, mientras el primer método implica recorrer todos y cada uno de los bits de la población ($popsize \times v \times k$ bits) el segundo tan solo vista algunos ($\boldsymbol{round}\{frmut \times popsize \times v \times k\}$ bits), lo que supone menor tiempo invertido en cada operación de mutación (que por poco que sea, es ganancia para el tiempo de convergencia). Cualquiera sea el método, en caso de emplear un criterio elitista, las soluciones consideradas élite no pueden ser alteradas por ésta operación.

El cambio de un bit produce una nueva solución, que puede estar "ubicada" en cualquier parte del espacio de soluciones. Así, la finalidad de la operación de mutación consiste en dar diversidad a la población y de ésta manera incentivar la exploración del espacio de diseño.

Llegados a éste punto, si ningún criterio de parada se cumple, se reinicia el ciclo. En relación a los criterios de parada, como se mencionó anteriormente, tres de ellos son: por baja diversidad de la población, alcanzar el límite de iteraciones y superar el límite de tiempo. Adicionalmente, se ha considerado como criterio de parada la permanencia de la misma solución como la mejor durante un número de iteraciones prolongado. Éste último es medido a través de la observación de la Desviación Estándar del MSE de la mejor solución durante n iteraciones (identificadas con la variable $iter\_\sigma_{MSE}$). Cualquiera sea el criterio de parada, la mejor solución de la última iteración es considerada el óptimo y la solución al problema de diseño de filtros digitales.

2.3 Algoritmo de Aprendizaje Incremental Basado en Poblaciones Adaptativo (APBIL)

La estructura básica del algoritmo PBIL adaptativo implementado se presenta en el Esquema Algoritmo 3 (ver Figura 2-6). En principio, se inicializa el arreglo de probabilidades P, de manera tal que todas las soluciones tengan la misma probabilidad de ocurrencia. A partir de P se genera una población de soluciones individuales, que son valoradas para tomar de ellas la mejor de todas y utilizarla en la actualización de las mimas probabilidades. En la versión adaptativa, antes de

realizar dicha actualización, se ajusta el valor de la tasa de aprendizaje, en función de la Entropía (la cual es una métrica de cuán distribuidas o dispersas se encuentran las probabilidades dentro del arreglo mismo). Finalmente, con la mejor solución y la tasa de aprendizaje a la mano, se actualiza P; aumentando las probabilidades de ocurrencia de las características que aparecen en las mejores soluciones de cada iteración y consecuentemente, reduciendo las probabilidades de las características que no hacen parte de ella. Si ningún criterio de parada se cumple, se repite el ciclo, retornando al segundo paso, el cual consiste en la generación de una nueva población de soluciones individuales para su valoración. De lo contrario, la búsqueda finaliza y la solución que es considerada el óptimo y solución al problema de diseño de filtros digitales planteado, será aquella cuyas características se destacan por poseer las mayores probabilidades de ocurrencia.

Figura 2-6: Esquema general del algoritmo APBIL.

Algoritmo 3: Algoritmo APBIL.

Entrada: An $M \times k$ probability array, called P.
Salida: An optimal design of a FIR or IIR digital filter.

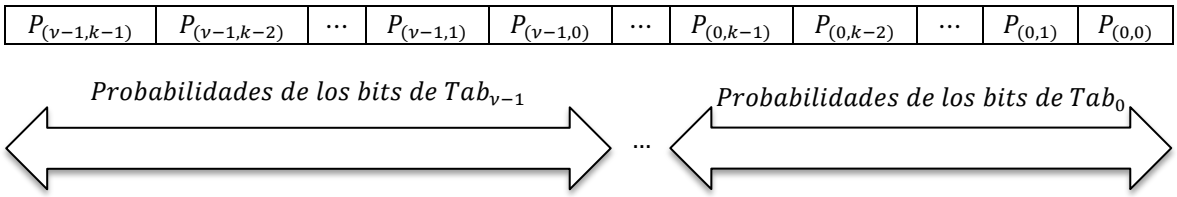
Inicion
$P_{(i,j)} = \frac{1}{2};\ \forall\ 0 \leq i \leq M-1\ \wedge\ 0 \leq j \leq k-1;$

Repetir
$Pop = Create\_Population(P);$
$Fitness = Evaluate\_Population(Pop);$
$Best = Choose\_Best(Pop, Fitness);$
$E = Entropy(P);$
$LR = Learning\_Rule(E);$
$P = Update\_Array(P, Best, LR);$
Hasta $(Stopping\_Criterion = false)$
Retornar $Best$;
Fin

Nombre de la fuente: Elaboración propia

Dada la estructura elegida para representar las soluciones (tal como se muestra en la Figura 2-1), la estructura del arreglo de probabilidades P es similar, con la diferencia de que en sus entradas, en vez de registrar los valores de cada bit, se almacenan las probabilidades de que el bit de la posición correspondiente sea igual a uno (1) (ver Figura 2-7). Así, $P_{(i,j)}$ representa la probabilidad de que el *j-écimo* bit, del *i-écimo tab*, sea igual a uno (1) y su complemento $\left(1 - P_{(i,j)}\right)$ se refiere a la probabilidad de que el bit en cuestión sea igual a cero (0).

Figura 2-7: Representación del arreglo de probabilidades del algoritmo APBIL.



Nombre de la fuente: Adaptado de [5]

Normalmente, cuando finaliza la búsqueda, las probabilidades se han cerrado tanto sobre una única solución que, es fácil distinguir sobre el mismo arreglo de probabilidades P la solución óptima. En éste punto, las probabilidades asociadas a la mejor solución son iguales o próximas a uno (1.0) o cero (0.0), y al redondear el valor de cada $P_{(i,j)}$ se obtiene el valor del bit respectivo, en cuyo caso, P se convierte en la solución óptima.

La inicialización del arreglo P está orientada a que todas las posibles soluciones posean igual probabilidad de ocurrencia, por lo cual las probabilidades se distribuyen de manera homogénea. Dado que la representación de la solución es binaria, se dispone de solo dos opciones por bit (0 ó 1), y por tanto las probabilidades se inician en 0.5.

La rutina ***Create_Population*** genera una población (de tamaño $popsize$) de soluciones individuales, a partir de las probabilidades registradas en P. Para cada solución, por bit se genera un número aleatorio en el rango cero a uno ([0, 1]), si dicho número es igual o menor que la probabilidad asignada al bit en cuestión, a éste se le asigna el valor de uno (1), de lo contrario, se le asigna el valor de cero (0).

La rutina ***Evaluate_Population*** valora la aptitud de las soluciones de la población para el problema de filtros digitales planteado (como se explicó en la sección 2.1 del presente capítulo). Según dicha valoración, la rutina ***Choose_Best*** selecciona la mejor de todas las soluciones de la población.

La rutina ***Entropy*** calcula la Entropía (E) de P, como se indica en la Ecuación (2-5). En un comienzo, cuando las probabilidades están homogéneamente distribuidas, la Entropía es máxima. Posteriormente, hacia el final del proceso de búsqueda, cuando las probabilidades se concentran alrededor de alguna solución, la Entropía tiende a cero. Este hecho permite definir el principal

criterio de parda del algoritmo. El cual consiste en asignar un límite o tolerancia al valor de la Entropía, cercano a cero. Así, la búsqueda finaliza cuando la Entropía Normalizada (E_n, razón entre la entropía calculada y la entropía máxima) desciende por debajo de tal valor.

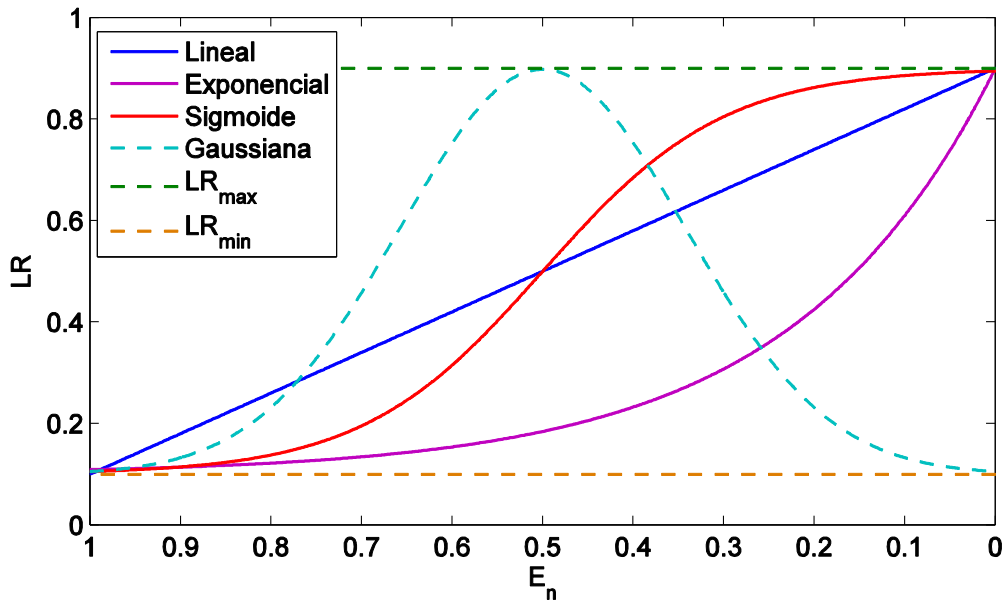
$$E = -\sum_{i=0}^{\nu-1}\sum_{j=0}^{k-1}\left[P_{(i,j)} \times \log\left(P_{(i,j)}\right)\right] \tag{2-5}$$

La tasa de aprendizaje (LR, *Learning Rate*) se actualiza por medio de la rutina $\boldsymbol{Learning\_Rule}$. Si la tasa de aprendizaje es constante, como ocurre el al algoritmo PBIL tradicional, valores pequeños de ésta (cercanos a cero) realizan una lenta actualización del arreglo de probabilidades, favoreciendo la *exploración* del espacio de soluciones, lo que a su vez mejora la calidad de la aproximación, a costa de largos tiempos de convergencia. Por otro lado, valores grandes favorecen la *explotación* de la información recolectada por soluciones halladas en iteraciones tempranas, reduciendo los tiempos de convergencia, pero sacrificando la calidad de la respuesta. Sin embargo, el algoritmo PBIL adaptativo (o APBIL) reúne los beneficios de ambas estrategias (exploración y explotación): Inicia con un valor mínimo de la tasa de aprendizaje (LR_{min}) para favorecer la exploración al principio del proceso de diseño y, conforme las probabilidades se van concentrando y la búsqueda se va definiendo, realiza incrementos graduales de la tasa de aprendizaje, hasta alcanzar su valor máximo (LR_{max}), acelerando cada vez más el tiempo de convergencia.

Como conclusión del proceso de sintonización del algoritmo APBIL (presentado en la sección 2.4.3), se observó que: El valor de la taza de aprendizaje mínima tiene influencia directa con la calidad de la respuesta o aproximación final; entre más pequeño sea su valor, mejor es la aproximación. Por otro lado, la tasa de aprendizaje máxima tiene influencia directa sobre el tiempo de convergencia; así, entre mayor sea su valor, más rápido finaliza el proceso de búsqueda. La función que la tasa de aprendizaje (LR) sigue para pasar de su valor mínimo (LR_{min}) al máximo (LR_{max}) se denomina regla de aprendizaje y depende del valor de la Entropía Normalizada (E_n). A continuación, se presentan las reglas de aprendizaje contempladas, su dinámica (ver Figura 2-8) y correspondientes ecuaciones (ver Ecuaciones de la (2-6) a la (2-9)).

Regla de aprendizaje: Especificación formal		
Lineal	$LR = LR_{max} - E_n \times (LR_{max} - LR_{min})$	(2-6)
Exponencial	$LR = LR_{min} + (LR_{max} - LR_{min}) \times e^{-4.5 \times E_n}$	(2-7)
Sigmoide	$LR = LR_{max} - \frac{LR_{max} - LR_{min}}{1 + e^{-10 \times (E_n - 0.5)}}$	(2-8)
Campana Gaussiana	$LR = LR_{min} + \frac{LR_{max} - LR_{min}}{\sqrt{0.32 \times \pi}} \times e^{-\frac{(4 \times E_n - 2)^2}{0.8}}$	(2-9)

Figura 2-8: Reglas de aprendizaje para el algoritmo APBIL.



Nombre de la fuente: Elaboración propia

Finalmente, la rutina ***Update_Array*** actualiza las probabilidades de P, por medio de una regla de Hebbian modificada (ver Ecuación (2-10)). Si uno de los bits (β) de la mejor solución de la población es igual a uno (1), la probabilidad asociada a dicho bit es incrementada. Por otro lado, si el bit es igual a cero (0), la probabilidad asociada se reduce.

$$P_{(i,j)_{New}} = \begin{cases} P_{(i,j)_{Old}} + \left(1 - P_{(i,j)_{Old}}\right) \times LR, & si\ \beta_{(i,j)} = 1 \\ P_{(i,j)_{Old}} \times (1 - LR), & si\ \beta_{(i,j)} = 0 \end{cases} \quad (2\text{-}10)$$

2.4 Proceso de sintonización

Dentro del proceso de diseño de un algoritmo, posterior a la etapa de implementación del mismo, se tiene la etapa de sintonización de sus parámetros (o variables de control), también conocida como *tuning*. En general todos los algoritmos de optimización metaheurística requieren de este tipo de ajustes para lograr un desempeño óptimo. En el caso particular del trabajo actual, se desea con él lograr reducir la variabilidad de los resultados y afectar positivamente el rendimiento del algoritmo (definido en términos de la calidad y el tiempo de ejecución):

- Desde un principio, la aleatoriedad de los algoritmos de optimización metaheurística y la infinidad de posibles soluciones que pueden ser formadas con las largas cadenas de dígitos binarios que conforman la representación numérica del filtro, hacen que sea poco probable obtener exactamente la misma solución cada vez que se repite el proceso de búsqueda, sin importar que se conserve la parametrización de todas las variables que controlan el algoritmo. Sin embargo, y como se expone en las secciones a continuación (ver secciones de la 2.4.1 a la 2.4.3), posterior a la etapa de sintonización se redujo la variabilidad de los resultados; así, en términos de la representación binaria, para un mismo problema de diseño se obtienen soluciones similares, que difieren en algunos de sus bits menos significativos, modificando ligeramente su aproximación al filtro deseado.

- El rendimiento, con frecuencia, es medido desde el punto de vista del cliente o usuario del algoritmo, a quien en general le interesa obtener resultados de calidad en el menor tiempo posible. Normalmente, ambos conceptos son antagónicos: La obtención de soluciones de alta calidad requiere aumentar el volumen de información y extender el tiempo de los procesos de exploración; aunque, es probable que en algún punto de la búsqueda la solución sea lo suficientemente buena y cálculos posteriores produzcan cambios poco significativos o ninguno, incurriendo en el uso excesivo de recursos computacionales y por ende, en largos tiempos de ejecución. Por otra parte, la obtención de resultados en corto tiempo no garantiza calidad en los mismos. La selección adecuada de los valores del conjunto de variables que controlan el algoritmo permite obtener un balance entre tiempo y calidad, según las necesidades del momento.

En síntesis, el proceso de sintonización desarrollado fue efectivo y alcanzó los objetivos planteados: El mejoramiento de las soluciones, la reducción de los tiempos de convergencia y la disminución de la variabilidad de los resultados fueron evidentes, al finalizar el proceso y también

paso a paso con cada nuevo ajuste realizado. Por esta razón, a partir de él se dedujo un esquema general de sintonización, el cual se expone a continuación, con la finalidad de esbozar la labor realizada con los algoritmos aquí referidos y que a su vez sirva de referencia al lector, de modo que sea aplicado como plan de sintonización de cualquier otro algoritmo de optimización metaheurística similar.

- Esquema general de sintonización

i. Ajustar las variables cuyos valores son conocidos; partiendo de la experiencia previa del diseñador sobre los beneficios de cierto rango de valores o por recomendaciones en la literatura.

ii. Asignar valores iniciales a las demás variables según se crea conveniente, en pro de facilitar su posterior ajuste. La asignación de valores debe ser dirigida a ubicar el algoritmo en un punto de operación en donde se garantice la calidad de los resultados, aunque el tiempo de convergencia sea considerablemente lento. Esto facilita la tarea de sintonización, ya que, en gran medida el esfuerzo se enfocará en la reducción de los tiempos de convergencia, con la menor afectación de la calidad de los resultados.

iii. Realizar y analizar las pruebas necesarias para el ajuste de las variables particulares del algoritmo. Si el número de alternativas resultantes, de la combinación entre los diferentes valores contemplados para las variables, es grande, se recomienda subdividirlas en pequeños grupos para facilitar su análisis. Posteriormente, las alternativas destacadas de cada sub agrupación serán contrastadas entre sí en nuevos grupos, hasta que salga a relucir la mejor entre ellas. En el caso de los algoritmos poblacionales, se recomienda que la primer subcategoría sea por tamaño de población.

iv. Realizar los ajustes finales a los criterios de parda; a partir de los resultados recopilados hasta este punto o de nuevas simulaciones en caso de ser necesario.

Además, con el fin de facilitar la depuración y reducción del número de alternativas a analizar, se recomienda determinar valores límite que permitan identificar qué resultados son aceptables y cuales no lo son. En el caso particular de las simulaciones realizadas para la sintonización, se

identificó que los MSE iguales a 0.0144, 0.0169 y 0.0225 servirían de fronteras para calificar los resultados de manera cualitativa en resultados buenos, aceptables, regulares y malos.

En cuanto al análisis, se prefirió la comparación gráfica de las alternativas. Sin embargo, cuando ésta no es concluyente se puede recurrir a una evaluación cuantitativa de las características que resaltan en cada serie de datos; como se muestra en el análisis de las últimas alternativas disponibles en la sintonización del GA (ve sección 2.4.2).
Para los tres algoritmos, las simulaciones elaboradas consistieron en aproximar el filtro calculado a un filtro ideal, comparados punto a punto en el domino de la frecuencia, a través del MSE entre los respectivos Espectros de Frecuencia, normalizados (ver Ecuación (2-4)). Para lo cual, se empleó una distribución homogénea de mil muestras de frecuencia. Sólo fueron contemplados diseños FIR, por ser más exigentes en relación al tamaño del filtro, y se evaluaron soluciones de 16 y 32 *tabs* por ser tamaños típicos. En cuanto a la representación binaria, se emplearon la representación de *punto fijo – complemento a dos* y la representación de *punto flotante – IEEE 754*, con precisiones de 32 y 64 bits por *tab*.

Finalmente, la sintonización recomendada para los algoritmos GA, TS y APBIL se muestra en las Tablas 2-9, 2-11 y 2-15, respectivamente (en total, se emplearon 400 simulaciones para la sintonización del GA, 120 para el algoritmo TS y 512 para el algoritmo APBIL). Adicionalmente, a lo largo del proceso de diseño se identificaron valores para las variables de control, que si bien son diferentes a la sintonización recomendada, están en la capacidad de favorecer el desempeño del algoritmo bajo condiciones diferentes a las presentadas en la sintonización final. En las Tablas 2-7 y 2-8 se presentan las alternativas a la sintonización del GA y en las Tablas 2-13 y 2-14 para el algoritmo APBIL.

2.4.1 Sintonización del GA

En un proceso de sintonización de cualquier algoritmo de optimización metaheurística, hay involucradas variables de control propias a la naturaleza del problema, otras debidas a la representación numérica y un tercer grupo que solo conciernen a la metaheurística seleccionada. Los dos primeros grupos son comunes entre los algoritmos aquí referidos; aunque no necesariamente comparten la misma sintonización. En lo referente al diseño de filtros digitales lineales e invariantes en el tiempo, las variables corresponden a los parámetros del filtro deseado, tales como el número de *tabs* (v) y el número de muestras en frecuencia, que se emplean para

definir la precisión de su representación en los planos del tiempo y la frecuencia, respectivamente. En relación a la representación numérica, la cual corresponde a una representación binaria, se consideran la precisión en bits (k) y el tipo de representación ($Optbr$), que pude ser de *punto fijo – complemento a dos* o *punto flotante – IEEE 754.*

Considerando lo mencionado hasta ahora y en relación al GA, las variables a sintonizar son: Representación binaria, activación o desactivación del criterio elitista (y el número de soluciones élite), tamaño de la población ($popsize$), el porcentaje o fracción de la población considerada el conjunto de las soluciones "buenas" ($frngood$), el método de selección de padres ($Optsp$), el número de puntos de cruce ($numcp$), el método de mutación ($Optmut$), el porcentaje o la fracción de mutación ($frmut$), y en relación a los criterios de parada: el porcentaje o fracción de la población que define el límite de mínima diversidad admisible ($frmindiv$), el número de iteraciones para la observación de la desviación estándar del MSE de la mejor solución ($iter\_\sigma_{MSE}$) y el límite de iteraciones ($iterlimit$).

- Ajuste inicial: sintonización del criterio elitista, del porcentaje de soluciones buenas y elección del método de mutación

Como se mencionó anteriormente, algunas variables pueden ser ajustadas de inmediato, según experiencia previa del diseñador sobre el beneficio de cierto rango de valores, o por recomendaciones de otros autores:

- El criterio elitista permite conservar inmutables un conjunto de soluciones (o una sola solución) consideradas como las mejores, obtenidas a lo largo de proceso de búsqueda [21]. Por el beneficio entregado, siempre se hará uso de éste criterio (en los tres algoritmos seleccionados). En el caso particular del GA, el número de soluciones élite seleccionado es de tres: Luego de la observación, iteración por iteración, de diversas pruebas realizadas durante la etapa de implementación, se encontró que usar menos de tres individuos élite genera problemas para propagar las características "genéticas" de dichos individuos, impidiendo escapar de los "óptimos locales". Por el otro lado, un número mayor de soluciones élite puede reducir la diversidad del conjunto de padres (destacándose aquellos casos en los que el método de selección favorece a los padres más aptos), diseminando con rapidez su información, reduciendo así la diversidad de la población misma y en consecuencia, se aumenta la probabilidad de que la búsqueda se estanque en un "mínimo local".

- Por recomendaciones de [8] y [21], la fracción de la población considerada como el conjunto de mejores soluciones será igual a 0.5. En otras palabras, el 50% de la población será remplazada (los menos aptos) por los hijos generados a partir del otro 50%.

- Como se mencionó en la presentación del GA (ver sección 2.2), se consideraron dos alternativas para el proceso de mutación, entre las cuales eligió el método de número de mutaciones fijas, dado que en comparación al método tradicional, demanda un número de operaciones mucho menor.

▪ Primer conjunto de pruebas: sintonización del número de puntos de cruce y evaluación de la representación binaria

En las simulaciones iniciales se realizaron variaciones en la representación binaria y cantidad de puntos de cruce (1, 2 y 3 puntos de cruce) para diferentes dos tamaños de problema (16 y 32 *tabs*). Las demás variables fueron fijadas en un punto donde se garantiza la calidad en los resultados, a costa de largos tiempos de simulación (ver Tabla 2-1). Los resultados de las pruebas son mostrados en la .

Tabla 2-1: Sintonización del GA: valores iniciales de los parámetros del algoritmo.

Parámetro	Valor asignado
$popsize$	32 soluciones
$Optsp$	Según *fitness*
$Optmut$	Número fijo de mutaciones
$frmut$	0.04
Criterios de parada	
Criterio de parada por baja diversidad	Inactivo
$iter\_\sigma_{MSE}$	200 iteraciones
$iterlimit$	20×10^3 iteraciones

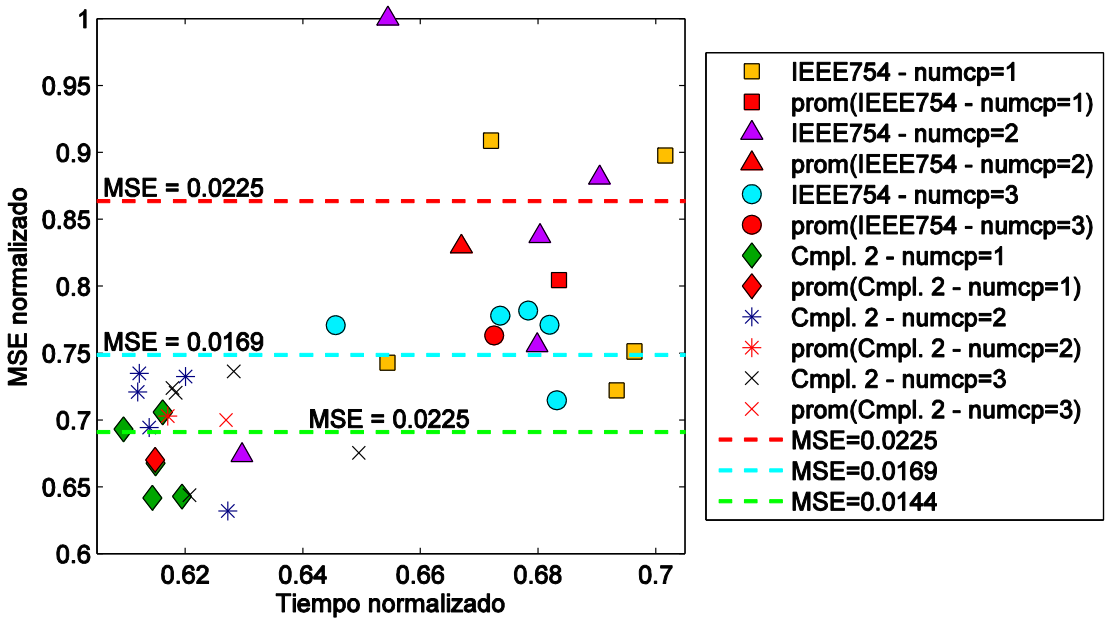
De los resultados se observa que:

- Sin importar la cantidad de puntos de cruce ni el tamaño del problema, la representación binaria de *punto fijo – complemento a dos* es superior en rendimiento a la representación binaria de *punto flotante – IEEE 754* (ver Figura 2-9 y Figura 2-10).

- Cuando el tamaño del problema es pequeño (16 *tabs*) existe mayor probabilidad de obtener mejores resultados con un solo punto de cruce. Mientras que, para problemas de gran tamaño (32 *tabs* o más), donde el número de bits empleados es mucho mayor, es preferible tomar más puntos de cruce.

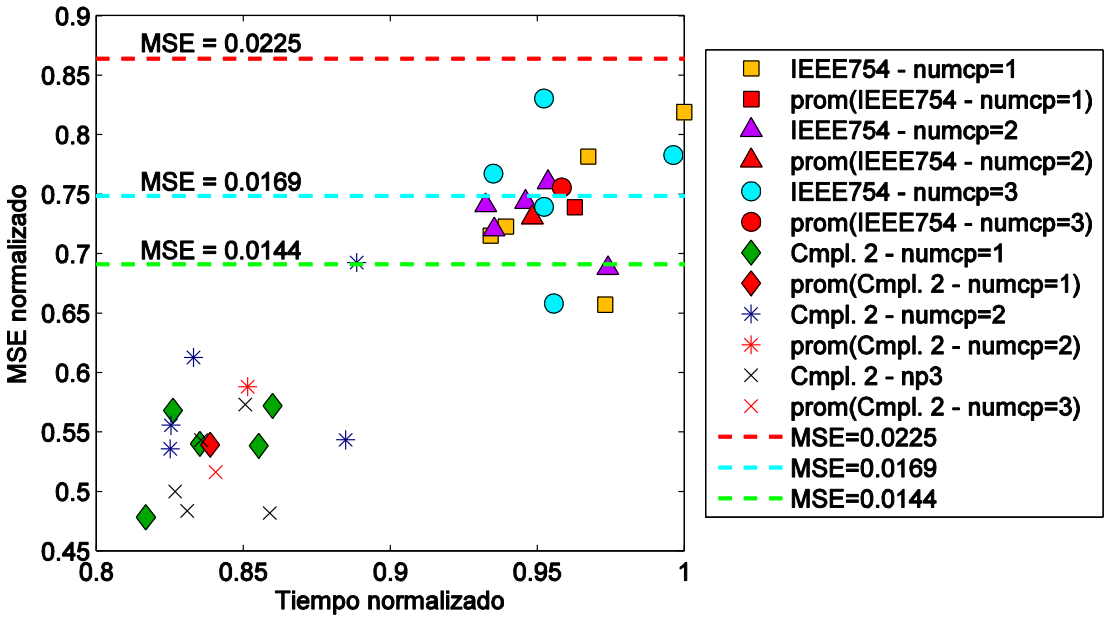
Por las razones mencionadas, la primera recomendación para la sintonización del GA es: De preferencia, emplear la representación binaria de *punto fijo – complemento a dos*. Sin embargo, sin importar la representación binaria elegida, cuando el producto $v \times k$ sea menor o igual a 768 bits utilizar un solo punto de cruce (un ejemplo de ello son los filtros digitales representados por 24 *tabs* con precisión de 32 bits por *tab*), y por el contrario, si dicho producto es mayor, emplear tres puntos.

Figura 2-9: Sintonización GA: representación binaria y puntos de cruce para filtros FIR de tamaño de 16 *tabs*.



Nombre de la fuente: Elaboración propia

Figura 2-10: Sintonización GA: representación binaria y puntos de cruce para filtros FIR de tamaño de 32 *tabs*.



Nombre de la fuente: Elaboración propia

Desde este punto del proceso de sintonización, se comienza a visualizar qué ajustes pueden ser adecuados para las variables de los criterios de parada:

- En adelante, se activa el criterio de parada por baja diversidad y se elige el 40% como límite de mínima diversidad admisible (después de haber probado con 30% y 35% sin obtener efecto alguno). Así, la búsqueda finalizará cuando más del 60% de las soluciones de la población sean idénticas.

- En relación al criterio de parada por observación de la desviación estándar el MSE de la mejor solución, se asume que 100 iteraciones son adecuadas (después de haber probado con 150 y 200 iteraciones sin obtener efecto alguno). De ésta manera, en caso de que la desviación estándar del MSE de la mejor solución en ese número de iteraciones, consecutivas, sea igual a cero, se asume que no habrán más cambios significativos y se finaliza la ejecución del algoritmo.

- Por otro lado, el límite de iteraciones pasará de 20 mil a 10 mil, debido a que se observó que en la mayoría de los casos, el MSE correspondiente al 1% de tolerancia del MSE final

es obtenido por debajo de éste último, e incluso, el promedio de iteraciones en las cuales se obtiene dicho valor de MSE es, por mucho, menor (1790 iteraciones).

- Segundo conjunto de pruebas: evaluación de los métodos de selección de padres, porcentajes de mutación y tamaños de población

En el siguiente grupo de simulaciones se realizaron variaciones del tamaño de la población (16, 32, 48 y 64 soluciones), del método de selección de padres (selección aleatoria y por peso según el *fitnes*) y del porcentaje de mutación (1% y 4%, valores mínimo y máximo recomendados por [21]), para soluciones de 16 *tabs*; resultando 16 combinaciones diferentes. Los resultados son mostrados desde la Figura 2-11 a la Figura 2-14.

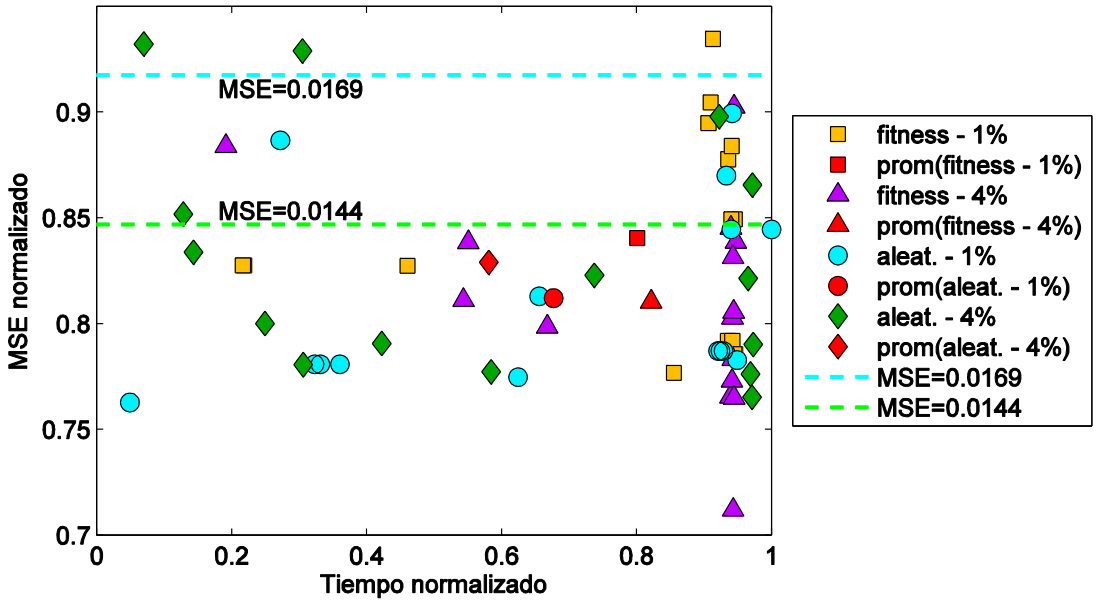
Inicialmente, los resultados fueron separados y analizados por tamaño de población. De entre cada grupo fueron seleccionadas las mejores alternativas, tomando como elementos de análisis el promedio, las ubicaciones de los valores máximo y mínimo (en ambos ejes) y la dispersión de cada serie de datos. Dando a cada criterio la relevancia según el orden mencionado, debido a que, en apariencia cada alternativa dentro de su agrupación es competitiva y presentan entre sí comportamientos similares (ver de la Figura 2-11 a la Figura 2-14). Se consideró que el promedio sería el elemento principal de análisis por tener la capacidad de almacenar dentro de sí la información de toda la serie e indicar la tendencia de los datos. Los demás criterios se consideraron de segundo nivel, sirviendo para hacer distinciones entre aquellas alternativas cuyos promedios se semejan.

Las alternativas destacadas por población se muestran en la Tabla 2-2 y desde la Figura 2-11 a la Figura 2-14 .

Tabla 2-2: Sintonización GA: alternativas destacadas por población.

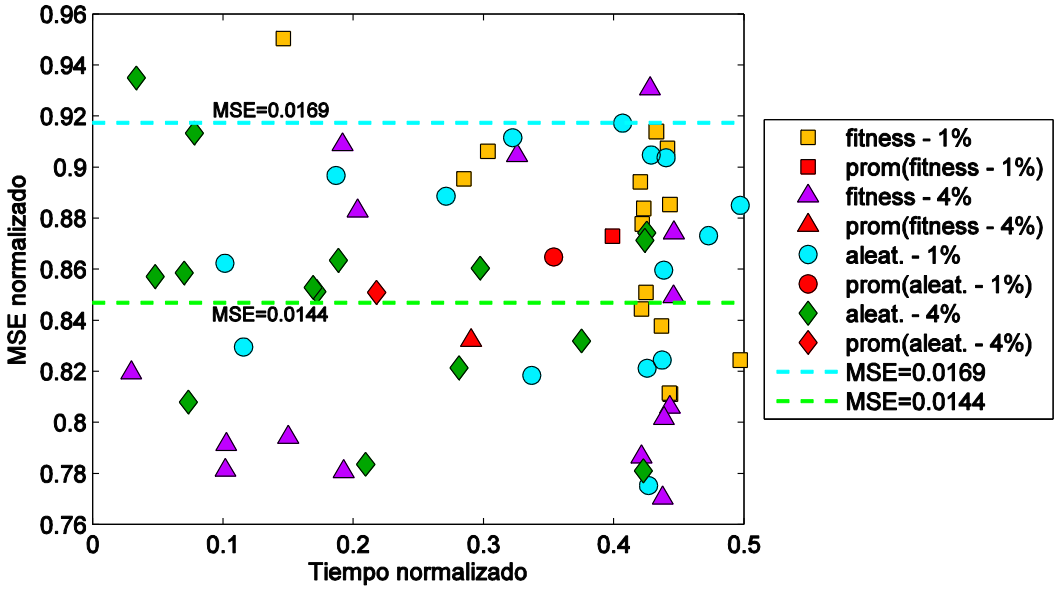
No.	Tamaño de la población ($popsize$)	Método de selección de padres ($Optsp$)	Porcentaje de mutación ($frmut \times 100$)
1	16	Selección aleatoria	4%
2	32	Por peso según el *fitness*	4%
3	48	Selección aleatoria	1%
4	64	Selección aleatoria	1%
5		Por peso según el *fitness*	4%

Figura 2-11: Sintonización GA: selección de padres y porcentaje de mutación para un tamaño de población de 64 soluciones.



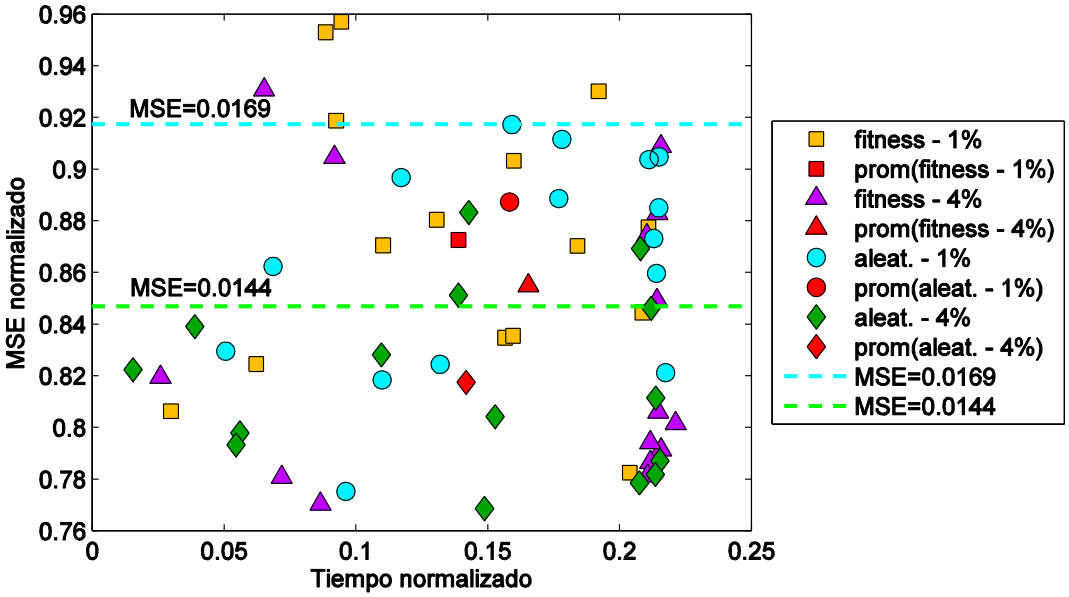
Nombre de la fuente: Elaboración propia

Figura 2-12: Sintonización GA: selección de padres y porcentaje de mutación para un tamaño de población de 48 soluciones.



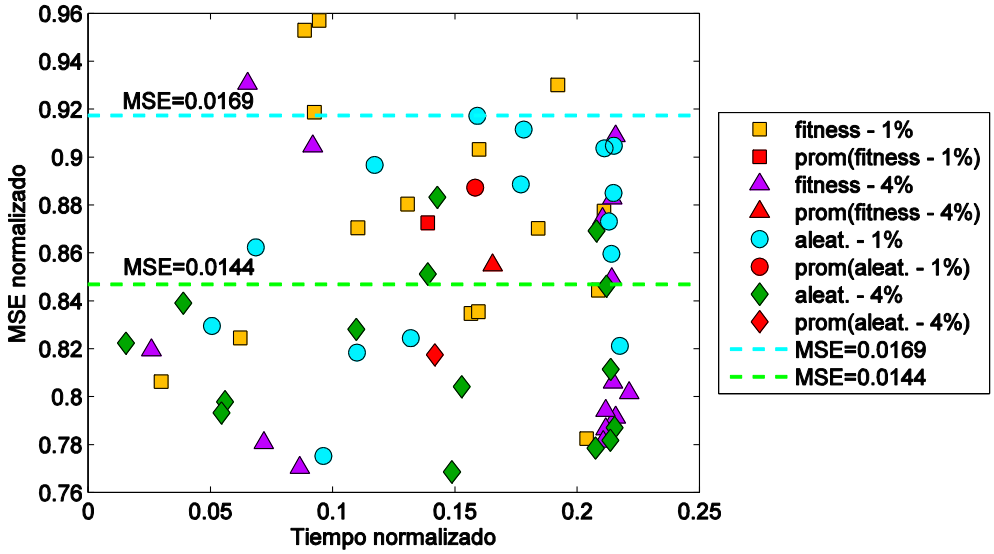
Nombre de la fuente: Elaboración propia

Figura 2-13: Sintonización GA: selección de padres y porcentaje de mutación para un tamaño de población de 32 soluciones.



Nombre de la fuente: Elaboración propia

Figura 2-14: Sintonización GA: selección de padres y porcentaje de mutación para un tamaño de población de 16 soluciones.



Nombre de la fuente: Elaboración propia

Al realizar la comparación entre las mejores alternativas (registradas en la Tabla 2-2) se observó:

- Respecto a la velocidad de convergencia, entre mayor sea el tamaño de la población, mayor es la probabilidad de obtener tiempos más lentos. Los tiempos más lento y más rápido registrados son de 8.194*s* y 530.160*s*, correspondientes a las poblaciones de 16 y 64 soluciones, respectivamente.

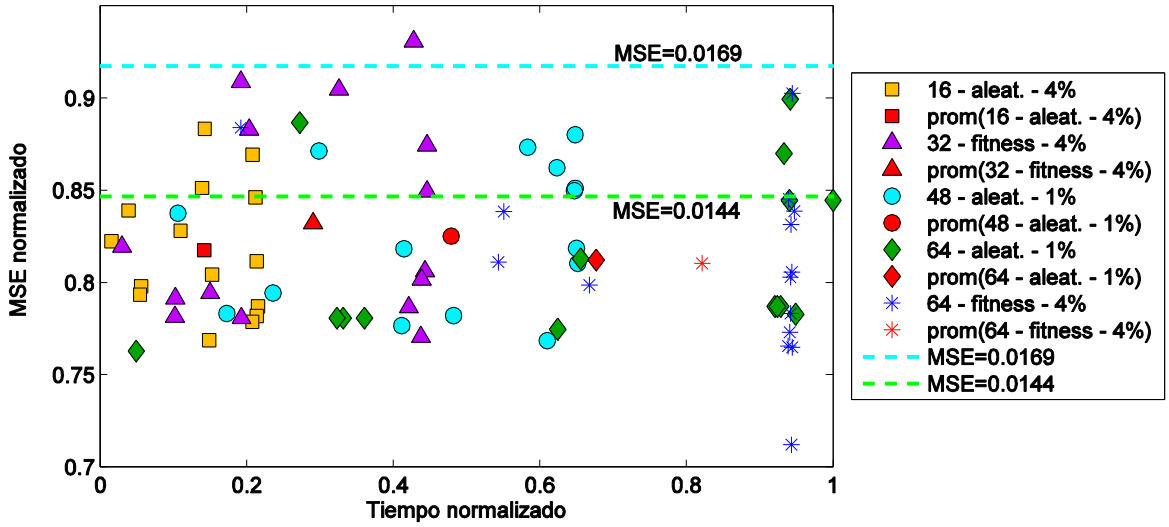
- Aunque todas las alternativas se muestran competitivas, tres se destacan por haber obtenido los promedios de MSE más bajos. Desde la mejor, éstas son: por la población de 64 soluciones, la selección por peso según el *fitness* con porcentaje de mutación del 4% (opción No.5 en la Tabla 2-2), y selección aleatoria con porcentaje de mutación del 1% (opción No. 4), y por la población de 16 soluciones, la selección aleatoria con porcentaje de mutación del 4% (opción No. 1).

- Existe la posibilidad de obtener buenas aproximaciones si se emplea una población grande con un límite de iteraciones menor, reduciendo considerablemente el tiempo de convergencia, dado que, aún se cumple que el mayor progreso se obtiene en iteraciones tempranas. La Tabla 2-16 en la cual se muestra el MSE correspondiente al 1% de tolerancia del MSE final para las dos alternativas consideradas por la población de 64 soluciones (opciones No. 4 y 5 en la Tabla 2-2), da indicios de la veracidad de dicha suposición.

▪ Tercer conjunto de pruebas (y pruebas complementarias): evaluación del límite de iteraciones

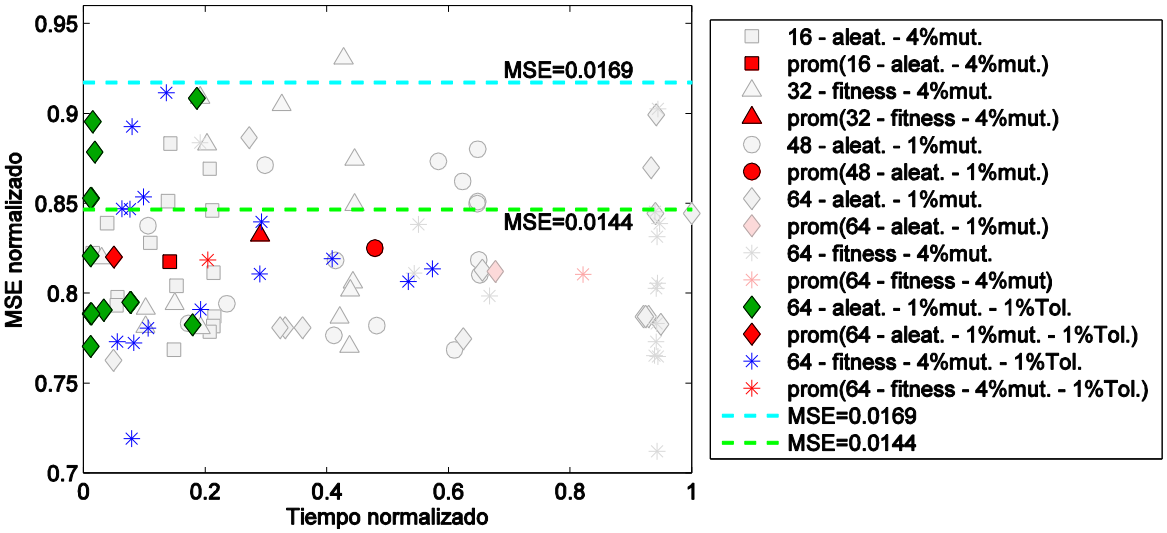
Continuando con la misma línea de análisis, el tercer conjunto de pruebas consiste en variaciones de límite de iteraciones, para las alternativas que ostentan la mejor calidad: Originalmente las series de datos correspondientes al tamaño de población de 64 soluciones poseen los mejores promedios respecto al MSE, pero presentan los tiempos más lentos (ver). Sin embargo, sus proyecciones, al 1% de tolerancia del MSE final, muestran que existe la posibilidad de obtener errores competitivos, en tiempos más cortos, que incluso pueden llegar a ser menores que los registrados para las alternativas anteriores (ver Figura 2-15 y Figura 2-16). Adicionalmente, se consideró la alternativa del tamaño de población de 16 soluciones (opción No. 1 en la Tabla 2-2), ya que ocupa el tercer puesto en calidad, según el promedio, y el primero en rapidez.

Figura 2-15: Sintonización GA: mejores alternativas, según el tamaño de la población.



Nombre de la fuente: Elaboración propia

Figura 2-16: Sintonización GA: MSE correspondiente al 1% de tolerancia del MSE final, para las mejores alternativas de la población de 64 soluciones.



Nombre de la fuente: Elaboración propia

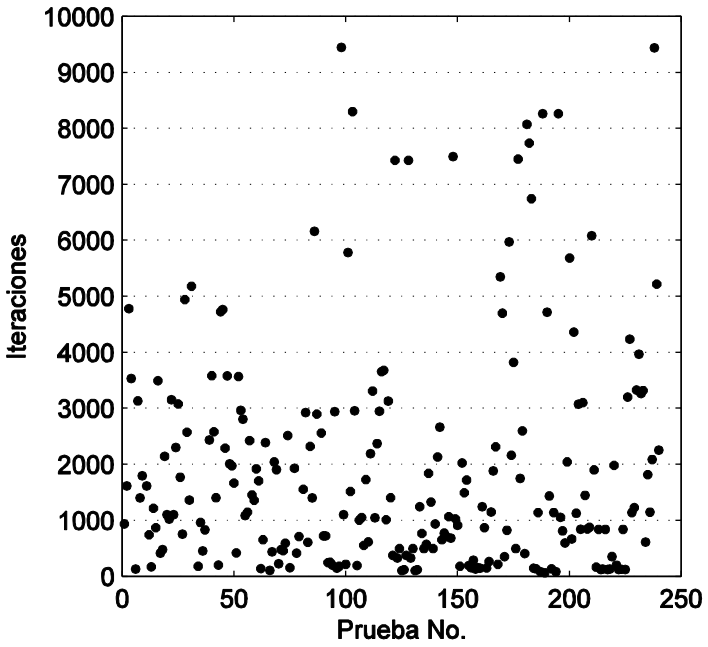
Inicialmente, se determinó qué límites podrían ser los más convenientes. Para lo cual se recurrió a la información de las iteraciones registradas para el 1% de tolerancia del MSE final (ver Figura 2-17), obteniendo la distribución porcentual mostrada en la Tabla 2-3, según la cual: el 87.88% de los casos se presentó por debajo de cuatro mil iteraciones y el 91.34% por debajo de 5 mil. Ambos

valores fueron considerados como límites para las nuevas pruebas, ya que cubren la mayoría de los casos y superan, por más del doble, el promedio de iteraciones correspondientes al 1% del MSE final (1906 iteraciones).

Tabla 2-3: Sintonización GA: distribución porcentual de las iteraciones correspondientes al 1% de tolerancia del MSE final.

Límite de iteraciones	Total de casos cubiertos	Distribución porcentual
≤ 1000	98	42.42%
≤ 2000	152	65.80%
≤ 4000	183	79.22%
≤ 4500	203	87.88%
≤ 5000	205	88.74%
≤ 6000	211	91.34%
≤ 7000	217	93.94%
≤ 6000	20	95.24%
≤ 8000	225	97.40%
≤ 9000	229	99.13%
≤ 10000	231	100.00%

Figura 2-17: Sintonización GA: distribución de iteraciones correspondientes al 1% de tolerancia del MSE final.



Nombre de la fuente: Elaboración propia

El análisis de alternativas se puede realizar de dos formas: con una comparación gráfica o con una evaluación cuantitativa de las características que resaltan en cada serie de datos. El análisis gráfico se prefiere ya que permite observar claramente el comportamiento de las series, pero si sus similitudes son numerosas y sus diferencias difícilmente percibidas, el análisis se hace complejo y depurar cuál es la mejor alternativa puede tardar, e incluso en el peor de los casos poner en riesgo la objetividad de la elección. Por otra parte, la segunda opción por sí sola no permite ver el comportamiento global, ya que muestra solo algunas características de las series, pero, permite definir previamente la forma de calificarlas y así determinar cuál es la destacada, sin recaer en subjetividades. Idealmente, se espera que ambos métodos lleguen a la misma elección.

En caso particular de la sintonización del GA, la evaluación gráfica de las últimas alternativas no permitió distinguir con claridad cuál es la más apropiada, por lo cual adicionalmente se ha recurrió al análisis por medio de una evaluación cuantitativa de las series de datos. A continuación se presenta el desarrollo y conclusiones de ambos análisis.

- Comparación gráfica de las mejores alternativas de sintonización

Se realizó la comparación gráfica entre las alternativas con 10 mil iteraciones como límite y sus análogas, con límite 4 mil, y se observó que:

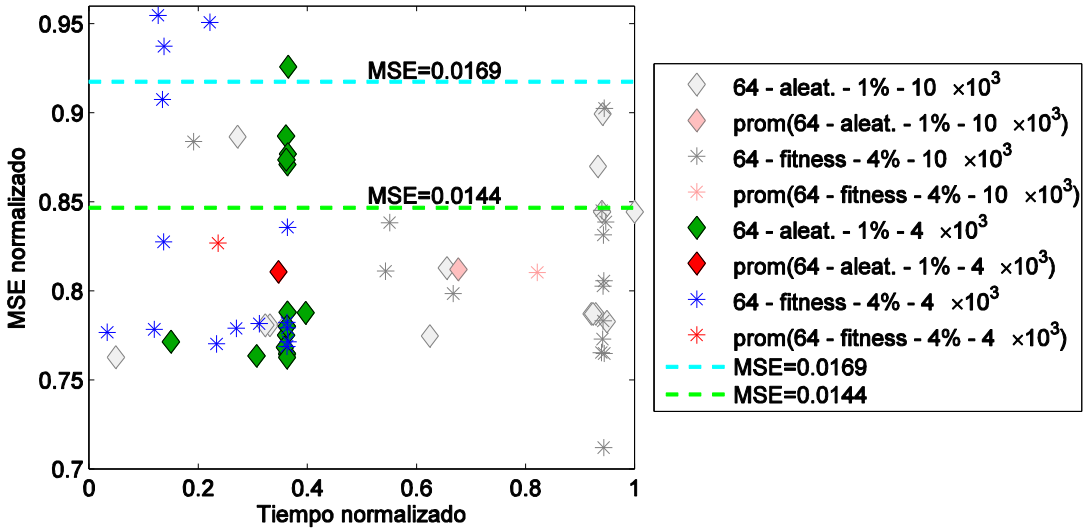
- Para las alternativas de 64 soluciones, al reducir el límite de iteraciones a 4 mil, se mejora el tiempo de convergencia sustancialmente, mientras sus errores continúan siendo competitivos (ver Figura 2-18).

- Por otro lado, la alternativa de 16 soluciones, con selección aleatoria de los padres y porcentaje de mutación del 4%, ve degradada la calidad de sus resultados al reducir el límite de iteraciones a 4 mil (ver Figura 2-19).

- Sin embargo, entre todas las alternativas, aquella con población de 16 soluciones y límite de 10 mil iteraciones continúa siendo la mejor opción (ver Figura 2-19).

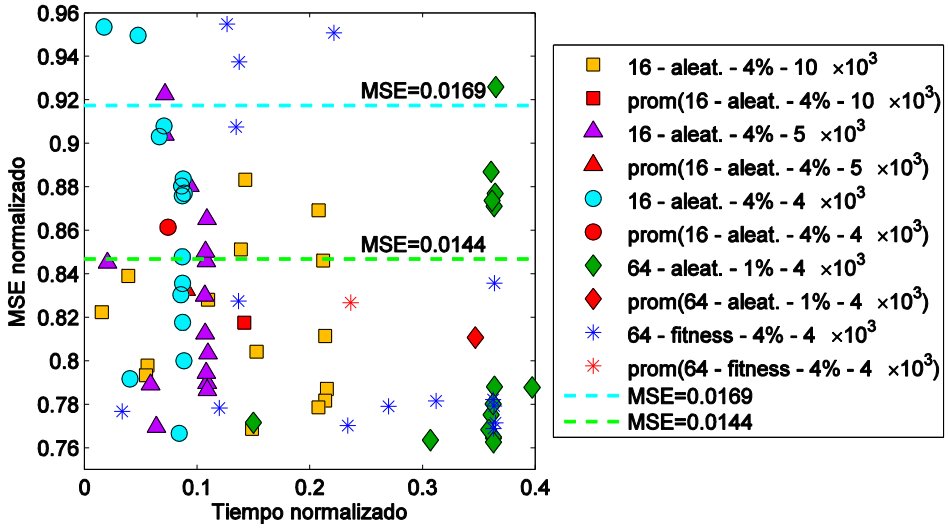
Así, después del análisis, solo sobrevive la alternativa con población de 16 soluciones, mas, el límite de 10 mil iteraciones, a ella asociado, aún se considera un número de iteración elevado. Por lo cual, se realiza una prueba adicional, con límite de 5 mil. Ésta última muestra ser más veloz y estar en la capacidad de brindar resultados satisfactorios, pero, al hacer un balance de lo que se observa a simple vista de los valores máximos en ambos ejes (tiempo y calidad), se puede llegar a dudar de cuál es la mejor elección: sus tiempos no superan el minuto, mientras los tiempos de la alternativa de 10 mil alcanzan hasta 4 minutos, y sus errores alcanzan a superar la barrera de $MSE = 0.0169$ (considerada la frontera entre soluciones “regulares” y “malas”), mientras la alternativa de 10 mil solo alcanza la mitad de la franja de lo “regular” (ver Figura 2-19). Ahora bien, si la decisión depende únicamente de un análisis gráfico, habrá que sopesar qué criterio pesa más y cuál será sacrificado, aun así, ambas alternativas están en capacidad de brindar soluciones satisfactorias.

Figura 2-18: Sintonización GA: mejores alternativas con tamaño de población de 64 soluciones y límites de 4 mil y 10 mil iteraciones.



Nombre de la fuente: Elaboración propia

Figura 2-19: Sintonización GA: mejores alternativas, bajo diferentes límites de iteraciones.



Nombre de la fuente: Elaboración propia

- Comparación por evaluación cuantitativa de las características para las mejore alternativas de sintonización

Una manera imparcial de disipar las dudas y tomar una decisión respecto a cuál alternativa es la más conveniente consiste en cuantificar las características de cada serie y llevar dichos valores a algún sistema de calificación que los integre. Esta evaluación se realizará a través de una suma ponderada de los siguientes factores de cada serie: valores máximos, mínimos, promedios y

deviaciones estándar de los datos de tiempo y MSE. En ella se castigarán los valores más altos y se premiará a los más bajos. Las series evaluadas corresponden a las mejores alternativas para la sintonización del GA, registradas en la Tabla 2-4 y Figura 2-19.

Tabla 2-4: Sintonización GA: alternativas finales.

No.	Tamaño de la población ($popsize$)	Método de selección de padres ($Optsp$)	Porcentaje de mutación ($frmut \times 100$)	Límite de iteraciones ($iterlimit$)
1	16	Selección aleatoria	4%	10000
2	16	Selección aleatoria	4%	5000
3	16	Selección aleatoria	4%	4000
4	64	Selección aleatoria	1%	4000
5	64	Según el *fitness*	4%	4000

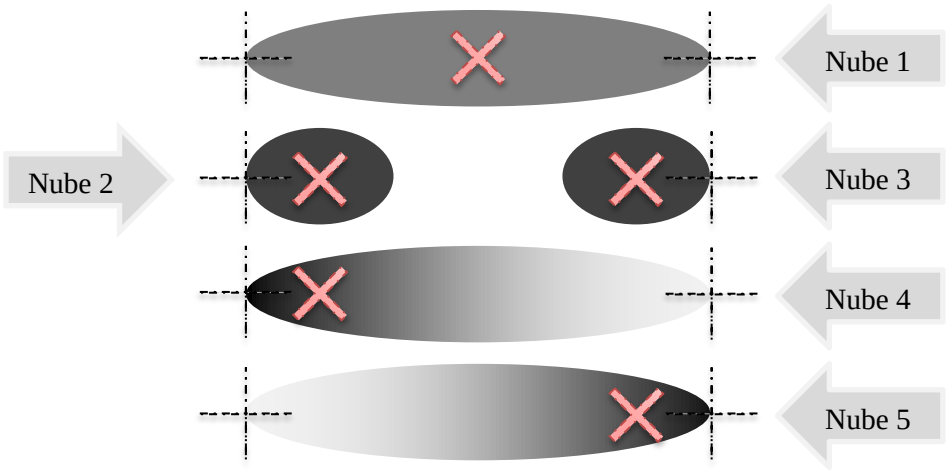
El primer paso consiste en realizar la calificación delos valores de cada factor de las series. Para ello, por factor, se han tomado los cinco valores, uno por serie, y su calificación ha sido aplicada de forma proporcional a su posición relativa respecto al máximo y mínimo de dicho grupo. La mejor calificación será uno y la pero cero. Así, la calificación $(C_{(i,j)})$ obtenida por el *i-ésimo* factor de la *j-ésima* serie de datos está definido como uno menos la razón entre el valor de interés $(v_{(i,j)})$ menos el valor mínimo (V_{min_i}) y el valor máximo (V_{max_i}) menos el valor mínimo, como muestra en la Ecuación (2-11).

$$C_{(i,j)} = 1 - \frac{v_{(i,j)} - V_{min_i}}{V_{max_i} - V_{min_i}} \tag{2-11}$$

Para saber qué peso será dado a cada factor, inicialmente se determinaran los pesos concernientes al tiempo, olvidándose temporalmente de los factores asociados al MSE. Imagine las series de datos como nubes de puntos e imagine cinco casos con las siguientes características (ver Figura 2-20): la nube uno será una nube grande, extensa, homogéneamente distribuida, cuyo promedio en el centro. Las nubes dos y tres serán nubes concentradas de extensión $1/3$ de la nube uno, cuyos promedios se encuentran en el centro de las mismas y cada una está apostada a extremos opuestos de la nueve uno, de manera que el mínimo de la nube dos coincide con el mínimo de la nube uno y por consiguiente sus promedio es menor y el máximo de la nube tres coincide con el máximo de la nube uno y por tanto su promedio es mayor. Ahora tendremos dos nubes más, tan extensas como la uno, coincidiendo sus máximos y mínimos, pero sus densidades de puntos se encuentran

recargadas a lados opuestos, así, la nube cuatro tendrá un promedio por debajo de la uno (no importa si se supone un promedio igual o distinto al de la nube dos) y la nube cinco tendrá un promedio mayor que la uno (sin importar si su promedio es igual al de la nube tres). Repasando y en relación a los promedios, las dos nubes de menor promedio son dos y cuatro, en el medio se encuentra la nube uno, y tres y cinco son las nubes de mayor promedio.

Figura 2-20: Ejemplo: series de datos vistas como nubes de puntos de diferente distribución.



Nombre de la fuente: Elaboración propia

El promedio y la desviación estándar reflejan la capacidad de obtener buenos resultados (o malos) y la confianza que se puede tener en ellos para poder obtener datos semejantes cada vez. Por otro lado, los valores mínimo y máximo enseñan qué tan lejos, en ambos extremos, puede llegar la serie, pero cada uno solo informa acerca de un único punto y no acerca de la tendencia. De lo anterior, se concluye que el promedio y la desviación estándar son los factores más importantes (por mucho) y los valores máximo y mínimo están en un segundo lugar. Entre el promedio y la desviación estándar la preferencia depende de a qué esquina se inclinan los promedios: si los promedios son opuestos respecto al punto medio, se preferirá siempre la de menor promedio sin importar la densidad de la nube. Por ejemplo, si se piensa en las nubes dos y cinco, se prefiere la dos, y entre las nubes tres y cuatro, se prefiere la cuatro. Pero, si los promedios se encuentran en posiciones cercanas la decisión dependerá de la desviación estándar: si los promedios se encuentran por arriba del punto medio, se preferirá a la que más dispersión tenga, dado que posee mayor probabilidad de alcanzar valores pequeños. Por el contrario, si los promedios están por debajo del punto medio, se preferirá a la nube más densa (que es la de menor desviación estándar), ya que brinda más garantías de obtener valores pequeños con mayor frecuencia. Así, al pensar en

casos como los las nubes dos y cuatro se prefiere la dos, y entre las nubes tres y cinco se prefiere la cinco.

Por las razones mencionadas, se ha decidido dar igual peso al promedio y a la desviación estándar e igual peso a los valores máximo y mínimo, de la siguiente manera: 35% será dado a los factores más relevantes (promedio y desviación estándar) y 15% a los otros (valores máximos y mínimos). Así, la suma de los pesos asignados a los diferentes factores relacionados a al tiempo dan como resultado a la unidad, como se muestra en la Ecuación (2-12), en donde $P_{t_{min}} = P_{t_{max}} = 0.15$ y $P_{t_{prom}} = P_{t_\sigma} = 0.35$. De forma análoga se realiza el análisis de la distribución de pesos para los factores relacionados al MSE, dando como resultado la misma asignación.

$$1.0 = P_{t_{min}} + P_{t_{max}} + P_{t_{prom}} + P_{t_\sigma} \tag{2-12}$$

Por otra lado, en cuanto a la relación rapidez y calidad, las cinco alternativas son competitivas respecto al MSE, pero sus tiempos de convergencia presentan diferencias marcadas; sin embargo, el tiempo más lento registrado entre las series evaluadas es de $229.44s$ (perteneciente a la alternativa con población de 16 soluciones y límite de 10 mil iteraciones), el cual es tolerable considerando el beneficio esperado (el diseño óptimo del filtro digital deseado). Así, considerando que ambos conceptos son relevantes en la determinación del rendimiento del algoritmo, pero, dejando a un lado el afán de obtener resultados en corto tiempo, se decide dar un poco de más importancia a la calidad por sobre la rapidez, de la siguiente manera: el 60% de la calificación será para los factores relacionados al MSE y el 40% restante a los factores que conciernen al tiempo de convergencia. Finalmente, la Ecuación (2-13) describe el método de calificación de cada serie.

$$\begin{aligned} C_j = {} & 0.6 \times \left(0.15 \times C_{MSE_{min}} + 0.15 \times C_{MSE_{max}} + 0.35 \times C_{MSE_{prom}} + 0.35 \times C_{MSE_\sigma}\right)_j \\ & + 0.4 \times \left(0.15 \times C_{t_{min}} + 0.15 \times C_{t_{max}} + 0.35 \times C_{t_{prom}} + 0.35 \times C_{t_\sigma}\right)_j \end{aligned} \tag{2-13}$$

Como resultado de la evaluación, prevaleció como la mejor alternativa la serie de población de 16 soluciones y límite de iteraciones 10 mil. En segundo lugar, y con una puntación próxima al primer puesto, está la serie de 16 soluciones y límite de 5 mil iteraciones. Individualmente, ambas series se destacaron en aspectos diferentes: la primera fue la mejor en relación al MSE, pero, la segunda estuvo entre los primeros puestos respecto al tiempo de convergencia. Como se esperaba, la evaluación cuantitativa permitió la elección de la mejor alternativa sin subjetividades, a pesar de

cuan reñida fuera la competencia. La información empleada y la calificación resultante se muestran en la Tabla 2-5 y en la Tabla 2-6, respectivamente.

Tabla 2-5: Sintonización GA: información empleada para la evaluación cuantitativa.

No.[1]	Tiempo [s]				$MSE \times 10^3$			
	min	$prom$	max	σ	min	$prom$	max	σ
1	8.194	75.206	229.440	38.041	11.8636	13.4186	16.7703	0.0242
2	10.869	47.827	58.085	14.311	11.8919	13.9182	17.0929	0.0426
3	9.241	39.261	47.020	11.655	11.8005	14.8991	18.2574	0.0614
4	79.506	183.985	210.635	30.360	11.6770	13.1982	17.2161	0.0666
5	17.794	125.244	193.378	60.919	11.8701	13.7249	18.3061	0.1056

[1]El número se relaciona con las series registradas en la Tabla 2-4

Tabla 2-6: Sintonización GA: evaluación cuantitativa.

No.[2]	Calificación tiempo					Calificación MSE					Total
	min	$prom$	max	σ	Subtotal	min	$prom$	max	σ	Subtotal	
1	1.00	0.75	0.00	0.46	0.58	0.13	0.87	1.00	1.00	0.82	0.724
2	0.96	0.94	0.94	0.95	0.95	0.00	0.57	0.79	0.70	0.56	0.715
3	0.99	1.00	1.00	1.00	1.00	0.42	0.00	0.03	0.45	0.23	0.536
4	0.00	0.00	0.10	0.62	0.23	1.00	1.00	0.71	0.39	0.74	0.539
5	0.87	0.41	0.20	0.00	0.30	0.10	0.68	0.00	0.00	0.25	0.273

[2]El número se relaciona con las series registradas en la Tabla 2-4

- Análisis de los criterios de parada

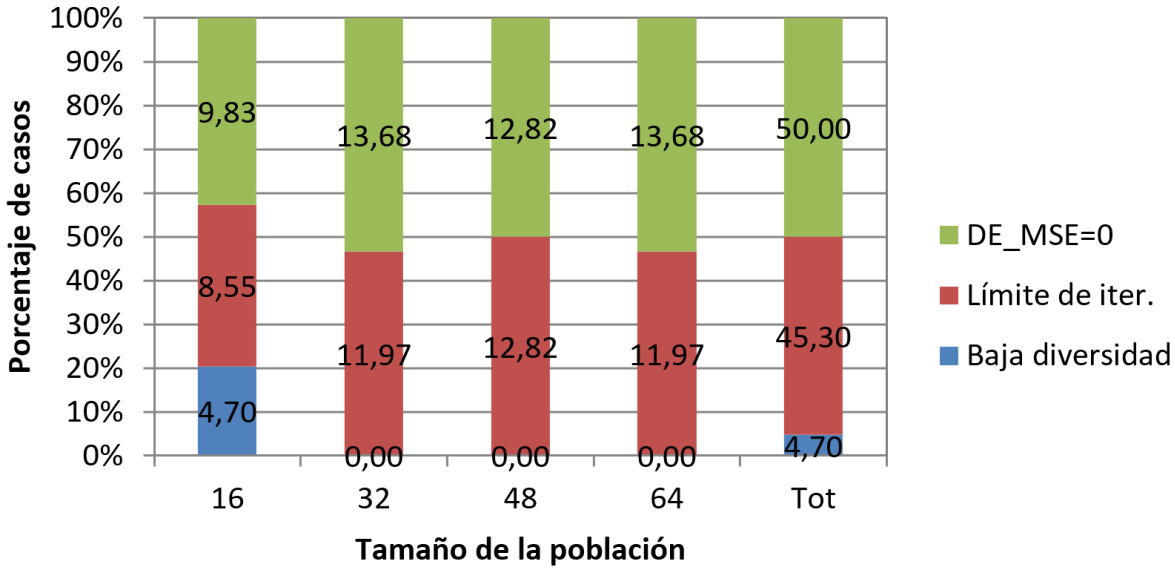
Adicionalmente, se analizó el porcentaje de participación de los criterios de parada dentro de los dos últimos grupos de pruebas y se observó que:

- El criterio de parada por baja diversidad de la población se presentó sólo cuando el tamaño de la población fue de 16 soluciones, representando el 4.7% del total de las pruebas (ver Figura 2-21).

- Dentro de las pruebas para el tamaño de población de 16 soluciones, el criterio de parada por baja diversidad sólo participó cuando el método de selección de padres y el porcentaje de mutación eran la selección por peso según el *fitness* y el 1%, respectivamente.

- Los demás criterios de parada tuvieron una participación equilibrada entre sí, en los diferentes escenarios: de forma global, en cada población (ver Figura 2-21), y bajo la implementación de las diferentes combinaciones del método de selección de padres y de los valores considerados para el porcentaje de mutación.

Figura 2-21: Sintonización del GA: participación global y por población de los criterios de parada.



Nombre de la fuente: Elaboración propia

Por lo anterior, se decide que el criterio de parada por baja diversidad deberá estar activo sólo cuando el tamaño de la población sea pequeño; para lo cual se ha considerado el rango de $(0, 24]$ soluciones. El motivo por el cual el criterio de parada sólo se activó bajo las circunstancias descritas se debe a que, sumado a una población pequeña, el método de selección de padres involucrado (selección por peso según el *fitness*) propaga rápidamente la información "genética" de los padres más aptos y reduce las probabilidades de participación en el proceso de apareamiento de los padres que no lo son, empobreciendo la diversidad de la población, y junto a ello, el bajo porcentaje de mutación (1%) no alcanza a insertar suficiente variedad en los "genes" de la población y en consecuencia, las mutaciones que no sobresalen dentro de la población son descartadas inmediatamente y aquellas que si lo hacen, no tiene la fuerza suficiente para estar dentro o cerca del grupo élite, reduciendo su posibilidad de propagación y permanencia en generaciones o iteraciones futuras.

- Sintonización recomendada para el GA

Finalmente, la sintonización recomendada para el GA que servirá al diseño de filtros digitales lineales e invariantes en el tiempo se expone en la Tabla 2-9. Adicionalmente, en las Tablas 2-7 y 2-8 se ofrecen alternativas de sintonización, pensadas en favorecer el rendimiento bajo condiciones diferentes a las presentadas en la sintonización final. Valores o combinaciones diferentes a las recomendadas no garantizan el rendimiento óptimo del algoritmo, ni el hallazgo de buenas soluciones.

Tabla 2-7: Sintonización alternativa GA: método de selección de padres y porcentaje de mutación para diferentes tamaños de población.

Tamaño de población $(popsize)$	Método de selección de padres $(Optsp)$	Porcentaje de mutación $(frmut \times 100)$
$(0, 24]$	Selección aleatoria	4%
$(0, 24]$	Por peso según el *fitness*	4%
$(40, Inf)$	Selección aleatoria	1%

Tabla 2-8: Sintonización alternativa GA: puntos de cruce según el total de bits de la solución.

Condición para el total de bits[3] $(v \times k)$	Número de puntos de cruce $(numcp)$
$v \times k \leq 768$ bits	1
De lo contrario	3

[3] k es la precisión en bits por *tab* y v es el total de *tabs* del filtro digital

Tabla 2-9: Sintonización recomendada para el GA.

Parámetro	Sintonización recomendada
Representación binaria ($Optbr$)	*Punto fijo – complemento a dos*
Tamaño de la población ($popsize$)	16 soluciones
Porcentaje de la población considerado de "buenas soluciones" ($frngood \times 100$)	50%
Método de selección de padres ($Optsp$)	Selección aleatoria
Número de puntos de cruce ($numcp$)	1 punto de cruce
Método de mutación ($Optmut$)	Número fijo de mutaciones
Porcentaje de mutación ($frmut \times 100$)	4%
Criterio elitista	Activo
Número de soluciones élite	3 soluciones
Criterios de parada	
Criterio de parada por $\sigma_{MSE} = 0$	Activo si $popsize \leq 24$
Iteraciones observadas para la evaluación de la σ_{MSE} de la mejor solución ($iter\_\sigma_{MSE}$)	100 iteraciones
Criterio de parada por alcance del límite de iteraciones	Activo
Límite de iteraciones ($iterlimit$)	5000 iteraciones
Criterio de parada por baja diversidad de la población	Activo
Porcentaje de mínima diversidad admisible ($frminDiv \times 100$)	40%

2.4.2 Sintonización del algoritmo TS

Los parámetros a ajustar para el algoritmo TS son: la representación binaria, la activación del criterio elitista, la fracción de bloqueos ($frlocks$), la activación de la Memoria Dinámica ($OptDM$), el tamaño máximo permitido para la Lista Tabú ($maxsizetl$), el tamaño mínimo del vecindario ($minNBHDsize$), y en cuanto a los criterios de parada: activación del criterio de parada por alcance del tamaño mínimo permitido para el vecindario ($OptminNBHDsize$), y el límite de iteraciones.

- Ajuste inicial: sintonización del criterio elitista y del tamaño mínimo admisible para el vecindario

Similarmente a lo ocurrido en el proceso de sintonización GA, los dos primeros pasos consisten en ajustar los parámetros cuyos valores son conocidos, partiendo de la experiencia previa o recomendaciones en la literatura, y asignar valores iniciales a las demás variables, según se crea conveniente en pro de facilitar su posterior ajuste:

- En relación al criterio elitista, como se mencionó en la sección anterior, por el beneficio entregado, siempre estará activo. En cuanto al número de soluciones élite, solo una será considerada como tal; porque a diferencia de lo que ocurre con el GA, el grupo élite no interviene en el proceso de búsqueda y sirve exclusivamente para no perder de vista a lo largo del proceso cuál fue el mejor vecino (o solución) y, por lo tanto, un número mayor de soluciones élite no brinda utilidad.

- Cómo se mencionó en la sección 2.1, el tamaño mínimo del vecindario tiene límites superior e inferior: El límite superior tiene lugar en la primer iteración, cuando ningún bit se encuentra bloqueado por la Memoria Atributiva, y es igual al total de bits de la representación del filtro digital ($k \times v$). Por otro lado, el límite inferior es determinado por el número máximo de bits que pueden ser bloqueados simultáneamente ($maxlocks$), como se muestra en la Ecuación (2-14). En ésta, el número máximo de bloqueos ($maxlocks$) es definido por la Ecuación (2-3), en función de la fracción de bloqueo ($frlocks$). Por lo anterior, y aunque el tamaño mínimo del vecindario puede ser deliberadamente fijado en un tamaño cualquiera entre estos límites, con el fin de aprovechar al máximo las ventajas brindadas por la AM, quien en suma, con todos los bloqueos y bits bloqueados, impide el retorno a soluciones ya visitadas y dirige la búsqueda a lugares no explorados, el tamaño mínimo del vecindario será determinado por su límite inferior, a través de la Ecuación (2-14).

$$minNBHDsize = k \times v - maxlock \tag{2-14}$$

- Primer conjunto de pruebas: sintonización de la fracción de bloqueo

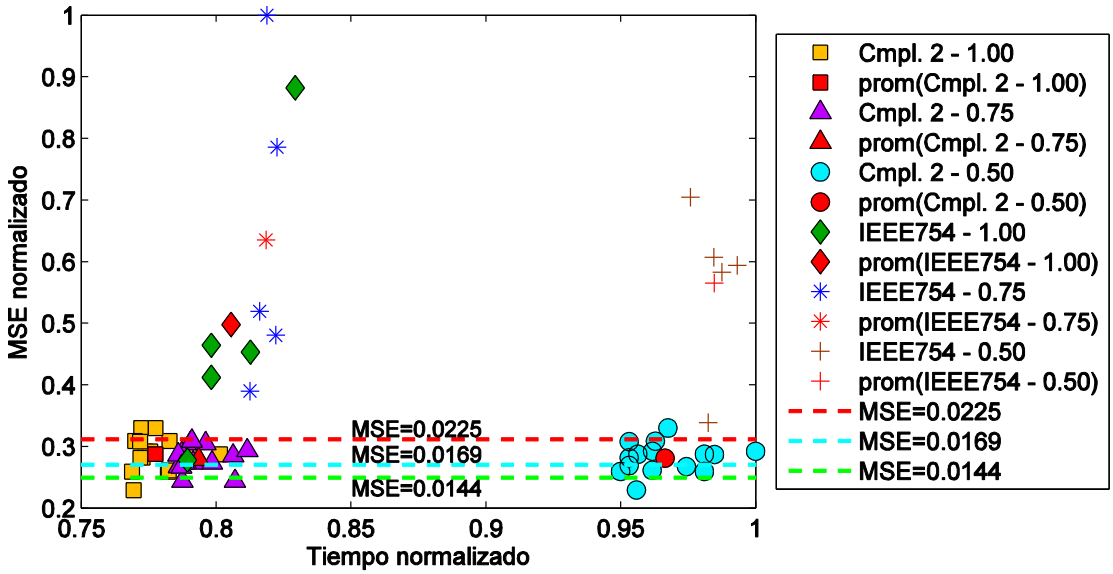
El primer conjunto de pruebas consiste en variaciones de la representación binaria y de la fracción de bloqueo (para la cual se evaluaron las fracciones de 1.0, 0.75 y 0.5), para soluciones representadas por 16 *tabs*. Con la introducción de variaciones de la fracción de bloqueo, el tamaño mínimo del vecindario también cambia para cada fracción (ver Ecuaciones (2-14) y (2-3)). Por esta

razón y con el fin de equilibrar el número de operaciones con el que cuentan las diferentes alternativas para entregar sus resultados, se desactivó el criterio de parada por alcance del tamaño mínimo del vecindario y el límite de iteraciones se fijó en 512. Este límite corresponde al punto en el cual al utilizar una fracción de bloqueo igual a la unidad, el mínimo tamaño del vecindario es alcanzado ($minNBHDsize = 2$ soluciones). En los demás casos, donde la fracción de bloqueo es menor, el correspondiente tamo mínimo del vecindario es alcanzado en iteraciones más tempranas, y con la medida implementada, la búsqueda continua aún después de alcanzarlo, hasta cumplir con el límite de iteraciones. Los valores iniciales de las demás variables se muestran en la Tabla 2-10 y los resultados de las pruebas en la Figura 2-22 y en la Figura 2-23.

Tabla 2-10: Sintonización TS: valores iniciales de los parámetros del algoritmo.

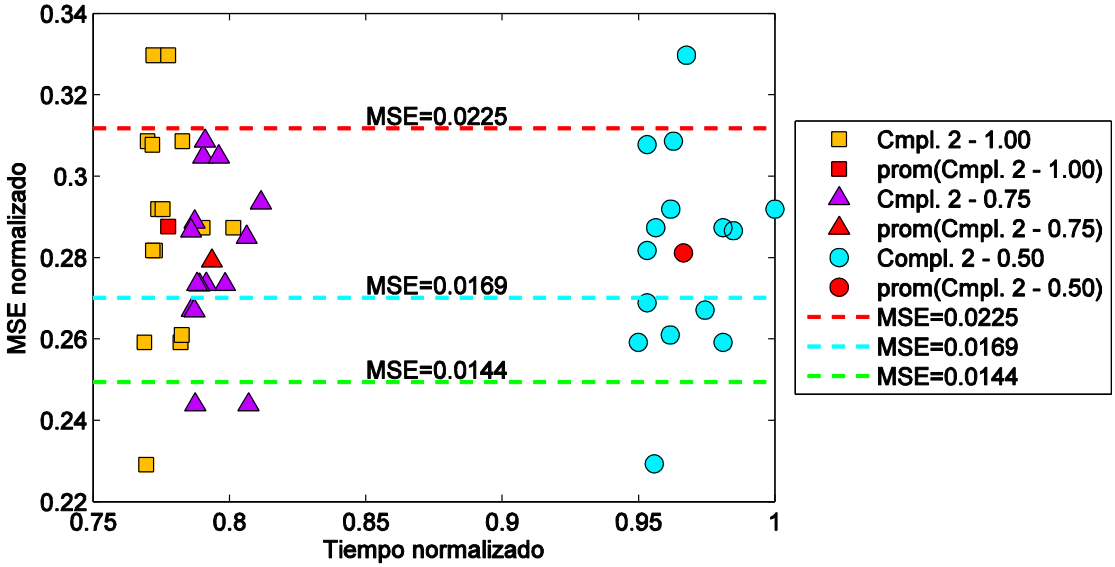
Parámetro	Valor asignado
Criterio de Memoria Dinámica ($OptMD$)	Activo
Tamaño máximo permitido para la TL ($maxsizelt$)	100000 soluciones
Criterios de parada	
Criterio de parada por alcance del mínimo tamaño del vecindario ($OptminNBHDsize$)	Inactivo
Criterio de parada por alcance del límite de iteraciones	Activo
Límite de iteraciones ($iterlimit$)	512 iteraciones

Figura 2-22: Sintonización TS: fracción de bloqueo y representación binaria.



Nombre de la fuente: Elaboración propia

Figura 2-23: Sintonización TS: fracción de bloqueo para la representación binaria *punto fijo – complemento a dos*.



Nombre de la fuente: Elaboración propia

Al observar los resultados presentados en la Figura 2-22 y en la Figura 2-23 se evidencia que:

- Las alternativas que involucren la representación binaria de *punto flotante – IEEE754* no ofrecen buenos resultados. La calidad es inferior a la ofrecida por la representación binaria de *punto fijo – complemento a dos* e incluso sus resultados están lejos de la frontera del $MSE = 0.0225$, revelando que las soluciones entregadas distan mucho de parecerse al filtro deseado.

- Los mejores resultados fueron obtenidos bajo la representación binaria de *punto fijo – complemento a dos.* Se ubican por debajo del $MSE = 0.0225$, en las áreas consideradas de resultados buenos y regulares.

- Las variaciones de la fracción de bloqueo no refleja ventajas respecto a la calidad en los resultados, pero si en cuanto al tiempo de convergencia. En apariencia, las series relacionadas con la representación binaria *punto fijo – complemento a dos* son semejantes entre sí, exceptuando el desfase en la línea del tiempo. La rapidez del proceso de búsqueda decrece proporcionalmente al valor de la fracción (algo evidente en ambos casos de la representación binaria).

De lo anterior se concluye que:

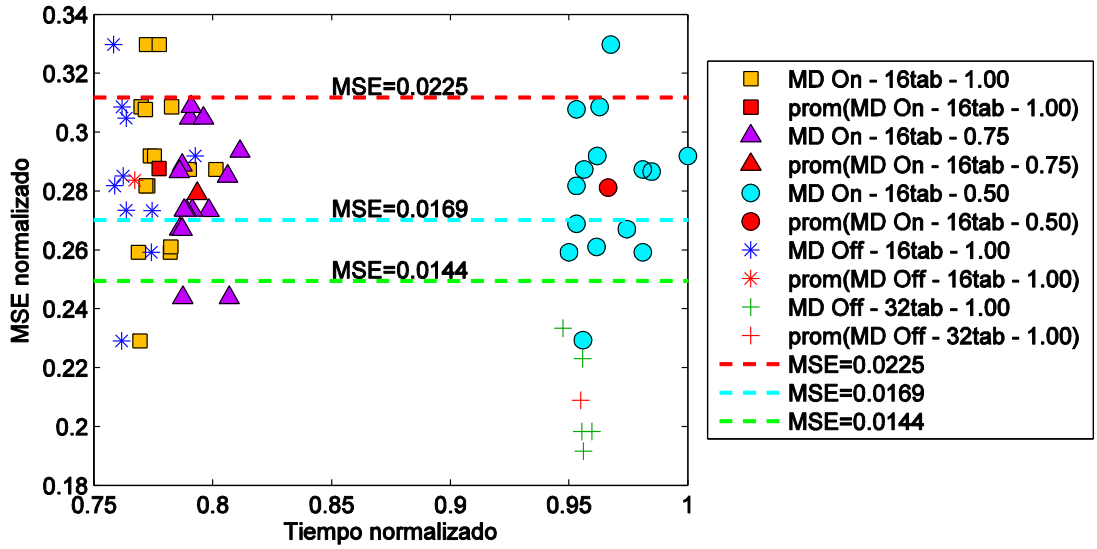
- La representación binaria de punto flotante (según el estándar IEEE 754) no es útil cuando el tamaño de la representación de la solución es pequeño (es decir, cuando se encuentra alrededor de los 16 *tabs* y precisión de 32 bits por *tab*).

- Variaciones de la fracción de bloqueo no afectan la calidad de los resultados, pero si el tiempo de convergencia de forma inversamente proporcional. Por lo cual, en adelante, se empleará una la fracción de bloqueo iguala a la unidad.

▪ Segundo conjunto de pruebas: sintonización de la Memoria Dinámica y representación binaria.

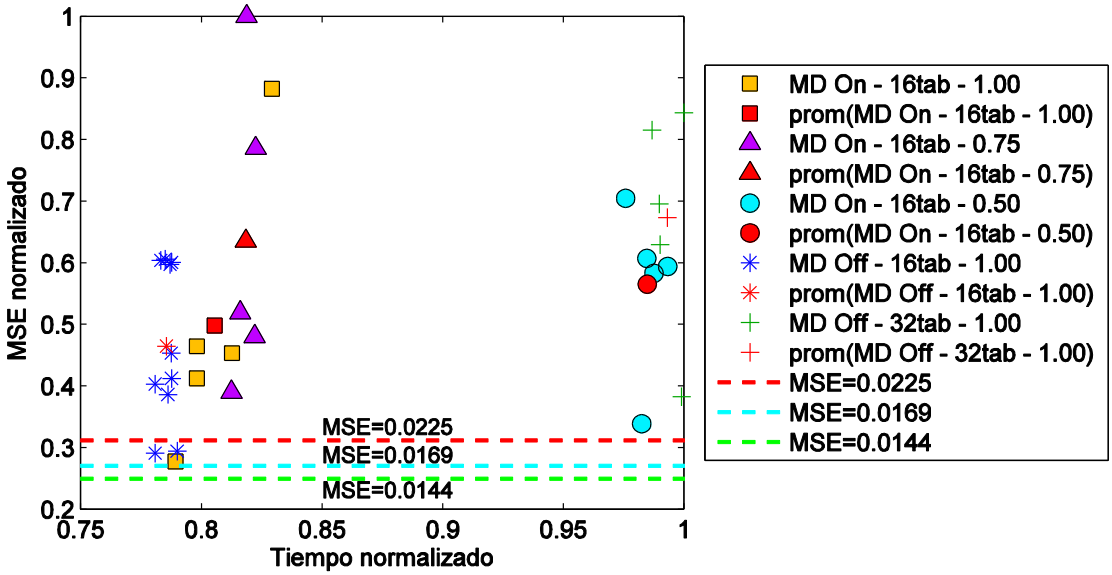
En el siguiente conjunto de pruebas se desactivo el criterio de Memoria Dinámica, es decir, no se limitó el crecimiento de la TL, y se realizaron variaciones de la representación binaria y del tamaño del problema (16 y 32 *tabs*). En la Figura 2-24 y en la Figura 2-25 se muestran los resultados, contrastados con los obtenidos en la prueba anterior, en donde al criterio de Memoria Dinámica estaba activo y el límite del tamaño de la TL era de 100 mil soluciones. Es de aclarar que en la gráfica, los tiempos presentados para los tamaños de 16 y 32 *tabs* están a diferentes escalas, la normalización de los tiempos fue hecha empelando los correspondientes valores máximos, y se hizo con la finalidad de poder visualizar con facilidad las diferencias en cuanto a calidad; en la escala real, los tiempos invertidos para la representación de 32 *tabs* es mayor que los tiempos de la representación de 16 *tabs*, con diferencias que van desde 10 a 36 minutos.

Figura 2-24: Sintonización TS: criterio de Memoria Dinámica para la representación binaria de *punto fijo – complemento a dos.*



Nombre de la fuente: Elaboración propia

Figura 2-25: Sintonización TS: criterio de Memoria Dinámica para la representación binaria *punto flotante – IEEE754.*



Nombre de la fuente: Elaboración propia

Normalmente se espera que la implementación del criterio de Memoria Dinámica reduzca el tiempo de cómputo, al no copar el espacio en memoria disponible, que pude ser requerido para otros cálculos; pero, como se muestra en la Figura 2-24 y en la Figura 2-25, la inactividad de éste criterio resultó ser más conveniente en relación al tiempos de convergencia. Al remitirse al código

escrito, ambas alternativas pueden ser definidas en una solo línea de código, sin embargo, dado el cómo fueron escritas, son muchas más las tareas ejecutadas internamente cuando el criterio está activo (sobre escribir la TL con la información más reciente de ella y la información nueva), en comparación con las que tienen que ejecutarse cuando está inactivo (ampliar el tamaño de la TL y agregar las soluciones nuevas). La sola sobre escritura conlleva tres tareas más: el almacenamiento temporal de los valores originales de la LT, mientras los valores viejos son borrados y los nuevos son escritos en el mismo espacio de memoria. Si se posee escasos recursos computacionales, cuando se repite la misma tarea en múltiples ocasiones la diferencia respecto al tiempo de convergencia se hace notoria.

Por otro lado, la representación binaria de punto flotante (según el estándar IEEE 754) nuevamente muestra ser poco apropiada para el algoritmo TS en el diseño de filtros digitales. Normalmente, se espera que el aumento del número *tabs* mejore la aproximación y por tanto, se reduzca el MSE, como ocurre con la representación binaria de *punto fijo – complemento a dos*, cuya mejoría fue significativa; no obstante, con la representación binaria de *punto flotante – IEEE754* pasó lo contrario: el aumento en el número de *tabs* produjo soluciones de pésima calidad, más erradas que aquellas concebidas con menos *tabs* (ver Figura 2-25).

- Tercer conjunto de pruebas: análisis del criterio de parada por alcance del límite de iteraciones

Se observó que, el comportamiento del MSE del mejor vecino, a lo largo del proceso iterativo, presenta un comportamiento similar la de una curva cuadrática, alcanzando su mayor decrecimiento dentro de la primer mitad del total de iteraciones y, en la otra mitad, aumenta hasta un punto semejante al de partida. En contraste, el comportamiento de la solución élite es similar en la primera mitad del total de iteraciones, pero, en la otra mitad, continúa constante hasta alcanzar el límite de iteraciones. El vértice descrito por la curva del MSE del mejor vecino es el punto de separación. Sólo en raras ocasiones, después de sobrepasado dicho punto, el mejor vecino aporta una nueva solución élite. Normalmente, el nuevo hallazgo no posee diferencias significativas respecto a la respuesta anterior, ocurre en un punto cercano al primer punto de separación y la segunda separación no tarda en aparecer, conservándose la apariencia de la curva cuadrática.

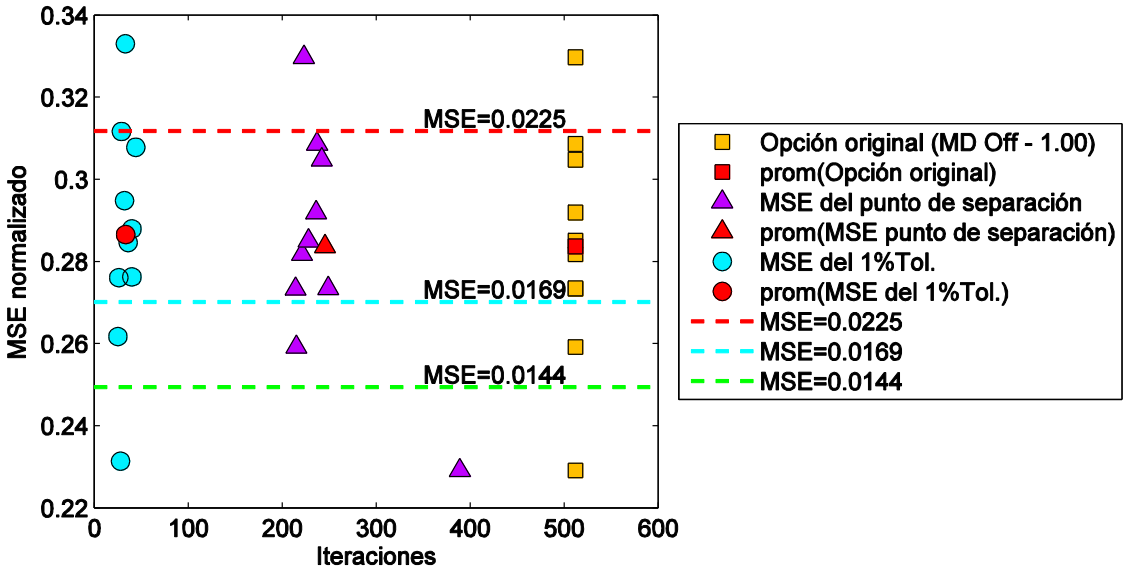
Existe dos alternativas, dentro de los criterios de parada, para aprovechar las características del comportamiento del MSE del mejor vecino en la reducción de los tiempos de convergencia: en primer lugar, reducir el límite de iteraciones hasta un punto próximo al punto de separación

definitivo entre la solución élite y el mejor vecino, y en segunda instancia, detener la búsqueda cuando en un número prudente de iteraciones consecutivas, el mejor vecino no produzca una nueva solución élite. Ambos criterios pueden operar simultáneamente. El último criterio de parada tiene cabida sólo sí se emplea el criterio elitista. Dado que los tiempos de convergencia para el algoritmo TS se incrementan considerablemente si el tamaño de la representación de la solución aumenta, el análisis a continuación se concentrara el determinación de un límite de iteraciones adecuado, y el uso del criterio de parada asociado a la producción de soluciones élite queda a libre elección del usuario.

En la Figura 2-26 y en la Figura 2-27 se muestran, para la mejor alternativa hallada hasta ahora, el MSE original de la serie, obtenido al final del límite de iteraciones (512 iteraciones), el MSE correspondiente al 1% de tolerancia del MSE final, y el MSE del punto de separación final, contra el número de iteraciones invertidas en cada caso. De los resultados, se infiere que:

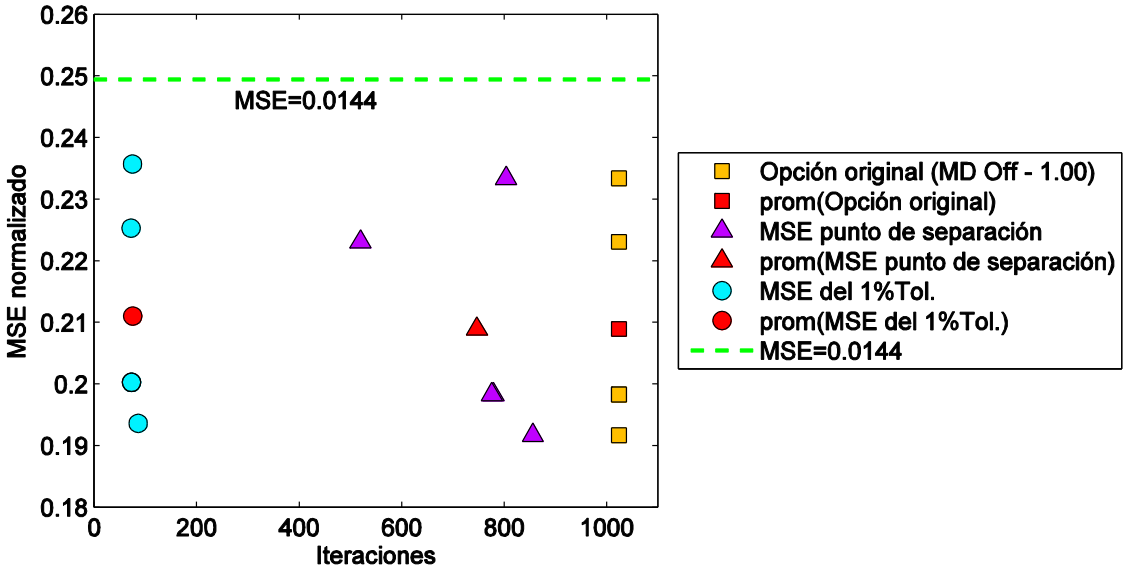
- El MSE correspondiente al 1% de tolerancia del MSE final es completamente competente en calidad y es obtenido en iteraciones tempranas, que no superan las 200 iteraciones.
- Es viable emplear un límite de iteraciones ubicado en un punto intermedio entre las iteraciones del MSE final y el MSE correspondiente al 1% de tolerancia, que esté cerca al punto de separación final, sin perder calidad y que permita reducir el tiempo de convergencia.

Figura 2-26: Sintonización TS: MSE vs. Número de iteraciones, para soluciones de 16 *tabs*



Nombre de la fuente: Elaboración propia

Figura 2-27: Sintonización TS: MSE vs. Número de iteraciones, para soluciones de 32 *tabs.*

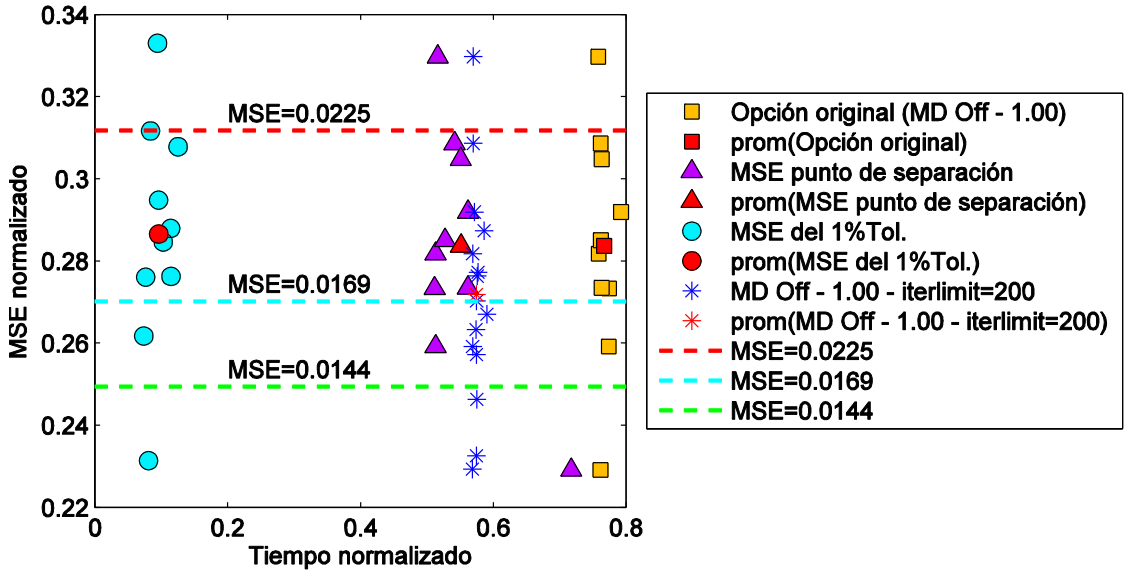


Nombre de la fuente: Elaboración propia

De lo anterior, se propone que 200 iteraciones pueden ser un límite adecuado que proporcione soluciones de calidad, en tiempos razones. En la Figura 2-28 se muestra la implementación del límite de iteraciones propuesto, demostrando que está en capacidad de brindar soluciones competentes en tiempos más cortos.

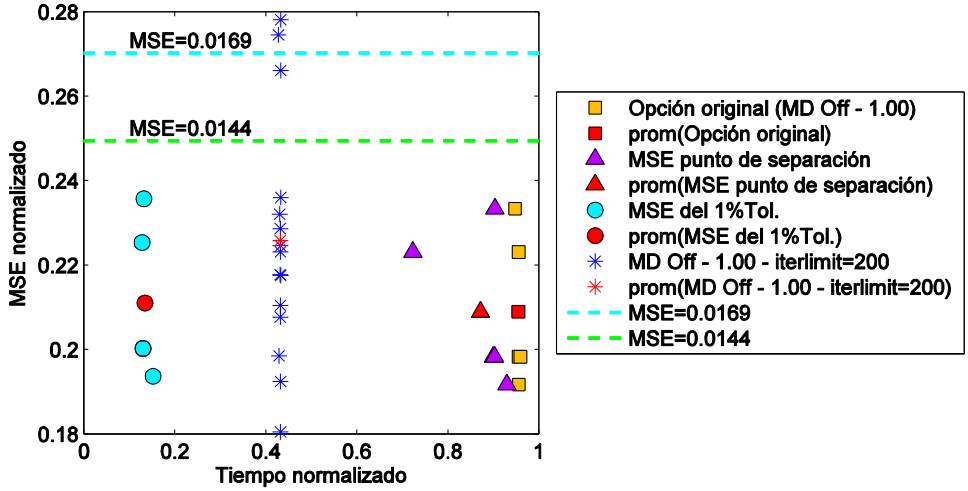
Es necesario aclarar que, los datos de tiempos mostrados en la Figura 2-28 y en la Figura 2-29, para el MSE del punto de separación y el MSE correspondiente al 1% de tolerancia, son tiempos estimados a partir del producto entre el número de soluciones visitadas a lo largo del correspondiente número de iteraciones (el cual es un dato conocido) y el tiempo invertido en los cálculos que involucran a una única solución (valor estimado a partir de la razón entre el tiempo de convergencia y el total de soluciones visitadas, que son datos conocidos). Por esta razón, los tiempos asociados a los errores mencionados, se muestran un poco desplazados a la izquierda de donde realmente deberían estar.

Figura 2-28: Sintonización TS: límite de iteraciones, para soluciones de 16 *tabs*.



Nombre de la fuente: Elaboración propia

Figura 2-29: Sintonización TS: límite de iteraciones, para soluciones de 32 *tabs*.



Nombre de la fuente: Elaboración propia

Aunque el criterio de parada por alcance del límite de iteraciones es suficiente por sí solo, por precaución, se reactivará el criterio de parada por alcance del mínimo tamaño del vecindario. Éste último servirá de apoyo, en caso fortuito de que el primero falle.

- Sintonización recomendada para el algoritmo TS

Finalmente, se han obtenido los ajustes pertinentes para los parámetros del algoritmo de TS concebido para el diseño de filtros digitales lineales e invariantes en el tiempo. Éstos se muestran

en la Tabla 2-11. Valores o combinaciones diferentes a las recomendadas no garantizan el desempeño óptimo del algoritmo, ni el hallazgo de buenas soluciones.

Tabla 2-11: Sintonización recomendada para el algoritmo TS.

Parámetro	Sintonización recomendada
Representación binaria ($Optbr$)	*Punto fijo – complemento a dos*
Fracción de bloqueo ($frlocks$)	1.0
Criterio de Memoria Dinámica ($OptDM$)	Inactivo
Tamaño máximo de la TL ($maxsizetl$)	*Inf*
Tamaño mínimo del vecindario ($minNBHDsize$)	Según la Ecuación (2-14)
Criterio elitista	Activo
Criterios de parada	
Criterio de parada por alcance del tamaño mínimo del vecindario	Activo
Criterio de parada por alcance del límite de iteraciones	Activo
Límite de iteraciones ($iterlimit$)	200 iteraciones

2.4.3 Sintonización del algoritmo APBIL

Las variables a sintonizar para el algoritmo APBIL son: la representación binaria, activación del criterio elitista (y el número de soluciones élite), el tamaño de la población, la regla de aprendizaje, la tasa de aprendizaje mínima (LR_{min}), la tasa de aprendizaje máxima (LR_{max}), y en cuanto a los criterios de parda: la tolerancia de la entropía ($tolE$) y el límite de iteraciones.

- Ajuste inicial: sintonización del criterio elitista, del límite de iteraciones y de la representación binaria
  - Comenzando con el criterio elitista, éste estará activo y el número de soluciones élite será de una única solución. Similarmente a lo que ocurre en el algoritmo TS, el grupo élite no interviene en el proceso de búsqueda y su función consiste en no perder de vista a la mejor solución. Su uso es simplemente preventivo, dado que, rara vez el resultados entregado por el algoritmo APBIL sin el uso de éste difiere del obtenido cuando está activo (comprobado durante la construcción del algoritmo).

  - En cuanto a la representación binaria, remitiéndose a los resultados de las pruebas de sintonización de los algoritmos TS y GA (registradas en las Figuras 2-9, 2-10 y 2-22 a 2-25), los cuales señalan que la representación binaria de punto flotante (según el estándar

IEEE 754) no está en la capacidad para brindar soluciones de calidad, se decide dirigir el proceso de sintonización del algoritmo APBIL a la representación de punto fijo (complemento a dos).

- Respecto al límite de iteraciones, éste será de 20 mil. Su uso es simplemente preventivo, pensado para aquellos casos en que el algoritmo no converja. Por ejemplo, a lo largo del proceso de sintonización, se presentaron casos en los que el criterio del límite de iteraciones tuvo que intervenir, éstos estuvieron relacionados con la utilización de una la tasa de aprendizaje mínima igual a cero, para las reglas de aprendizaje lineal y constante. Este hecho se puede observar en la Figura 2-31 y en la Figura 2-32, en la cual todos los puntos vinculados al tiempo máximo le pertenecen. En general, el criterio de parada relacionado a la tolerancia de la entropía es el encargado de detener la búsqueda. Bajo éste criterio la iteración máxima percibida fue de 8688 iteraciones, que son menos de la mitad del límite fijado.

Debido a problemas de convergencia, las tasas de aprendizaje iguales a cero fueron sustituidas por 1×10^{-7}. Éste valor, próximo a cero, representa el límite inferior para las tasas de aprendizaje dentro del rango (0, 1]. Los casos mencionados en el párrafo anterior ya contaban con ésta modificación. En adelante, al igual que en el caso anterior, cuando se mencione la tasa de aprendizaje cero será para referirse al límite inferior (1×10^{-7}).

- Primer conjunto de pruebas: sintonización de la tolerancia de la entropía

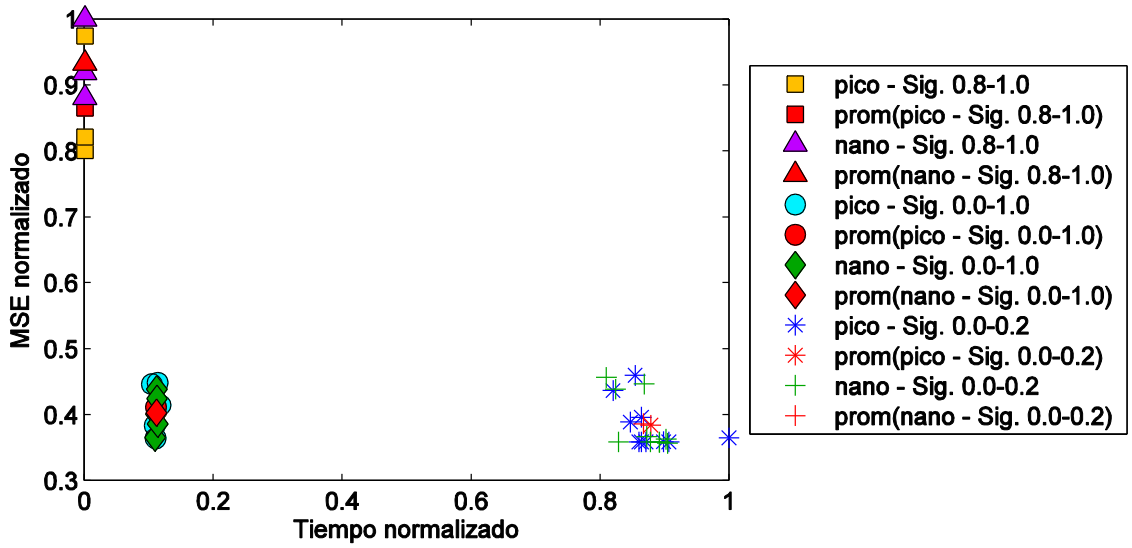
El primer conjunto de pruebas estuvo dirigido a determinar el valor apropiado para la tolerancia de la entropía. Para ello se seleccionó una regla de aprendizaje cualquiera (en éste caso la regla sigmoide), se evaluaron dos precisiones diferentes de tolerancia (1×10^{-9} y 1×10^{-12}) y se hicieron variaciones de las tazas de aprendizaje mínima y máxima (conformando las parejas (0.0, 0.2), (0.0, 1.0) y (0.8, 1.0)). Las parejas de tasas de aprendizaje fueron seleccionadas pensando en cubrir situaciones extremas: la primer pareja (0.0, 0.2) favorece la exploración del espacio de soluciones por sobre la explotación de la información recolectada por las soluciones halladas en iteraciones tempranas, la pareja (0.8, 1.0) propende por la explotación por sobre la exploración, y la pareja (0.0 y 1.0) intenta equilibrar ambos conceptos. Como se mencionó en la sección 2.3, la exploración del espacio de diseño favorece la calidad de los resultados, por otra parte, la explotación de las soluciones halladas busca la reducción de los tiempos de convergencia, y ambos

conceptos son contrapuestos; la sintonización idónea será aquella que combine apropiadamente ambos objetivos. En la Tabla 2-12 se registran los valores iniciales de las demás variables y en la Figura 2-30 se muestran los resultados de las pruebas.

Tabla 2-12: Sintonización APBIL: valores iniciales de los parámetros del algoritmo.

Parámetro	Valor asignado
Representación binaria ($Optbr$)	*Punto fijo – complemento a dos*
Tamaño de la población ($popsize$)	64 soluciones
Criterio elitista	Activo
Número de soluciones élite	1 solución
Criterios de parada	
Criterio de parada por alcance del límite de iteraciones	Activo
Límite de iteraciones ($iterlimit$)	200 iteraciones

Figura 2-30: Sintonización APBIL: tolerancia de la entropía.



Nombre de la fuente: Elaboración propia

De los datos mostrados en la Figura 2-30, se observa que:

- Sin importar los valores o amplitud entre las tasas de aprendizaje mínima y máxima, cambios en la precisión de la tolerancia de la entropía no representan variaciones significativas.

- La única diferencia perceptible a simple vista se da en la pareja (0.8, 1.0), para la cual se observa una pequeña pérdida de calidad al usar una precisión nano (1×10^{-9}). Sin embargo, de antemano se conoce que, esta pareja sacrifica los buenos resultados por la rapidez y por ende, no es un argumento de peso para tomar una decisión.

Así, debido a que los valores propuestos para la tolerancia de la entropía no generan diferencias perceptibles en la calidad de los resultados, y en apariencia tampoco lo hacen con el tiempo de convergencia, y que además, se sabe que procedimentalmente después de haber alcanzado la precisión nano, para llegar al a precisión pico (1×1^{-12}), se requiere de cálculos adicionales, que a su vez demandan un tiempo extra (aunque en la práctica imperceptible), se opta por sintonizar la tolerancia de la entropía en la precisión nano.

Cabe resaltar que en este punto el número de variables faltantes por sintonizar se ha reducido a la mitad, y es que en general, el número total de variables del algoritmo APBIL es pequeño; pero, la dificultad de su sintonización radica en la infinidad de valores que pueden ser seleccionados para las tasas de aprendizaje dentro del rango $(0, 1]$ cuando no se conoce algún conjunto de valores recomendados, y conjuntamente con las posibles reglas de aprendizaje el número de combinaciones a elegir es igualmente infinito. En principio, el rango mencionado puede ser discretizado para reducir las posibilidades; para el algoritmo diseñado, se han contemplado valores de tasas de aprendizaje que van de cero a uno, en incrementos de 0.2, y las reglas de aprendizaje *exponencial*, *sigmoide*, *lineal* y *constante* (esta última refiere al algoritmo PBIL tradicional). Resultando un total de 51 combinaciones, sin contar con las variaciones que se pueden hacer al tamaño de población. Con el objetivo de reducir el número de posibilidades, las pruebas para la elección del tamaño de la población, de la regla y las tasas de aprendizaje, fueron realizadas como se describe a continuación:

- Segundo conjunto de pruebas (1ro de la segunda etapa del proceso de sintonización): evaluación de reglas y tasas de aprendizaje, para un tamaño de población de 64 soluciones

Para las primeras pruebas de la segunda etapa se seleccionó el tamaño de población más grande contemplado, correspondiente a 64 soluciones, por ser la condición más conveniente para las 51 alternativas: entre mayor sea el número de soluciones, mayor es la probabilidad de que una solución buena parezca en la población. En las Figuras 2-31 a 2-34 se puede apreciar los resultados y el comportamiento general de las diferentes reglas de aprendizaje ante variaciones de las tasas.

De entre las 51 combinaciones, fueron seleccionadas las alternativas que tuvieron al menos un resultado por debajo del $MSE = 0.0225$ para ser parte del siguiente conjunto de pruebas, en donde el tamaño de la población fue reducido a 48 soluciones. Aquellas que cumplen con el criterio de eliminación son: por la regla exponencial las parejas de tasas de aprendizaje en las que se involucran las tasas cero y 0.2, aportando un total de nueve alternativas, por la regla sigmoide aquellas parejas que involucran la tasa cero y aquellas cuyo valor mínimo es 0.2 y su valor máximo está entre 0.4 y 0.8, incluyéndoles, para un total de ocho alternativas, por la regla lineal sólo la opción (0.2, 0.4) y por la regla constante la tasa 0.2. En total son 19 las alternativas sobrevivientes.

Las alternativas descartadas fueron todas aquellas cuyas tasas de aprendizaje mínima (o única tasa, en el caso de la regla constante) superó el valor de 0.2. Esta condición permite afirmar que: el grupo de alternativas que favorecen la rapidez por sobre la calidad son todas aquellas en las que la taza de aprendizaje mínima (o única tasa, en el caso de regla constante) es mayor o igual a 0.4 (considerando la discretización hecha al rango de valores posibles para las tasas de aprendizaje).

- Tercer conjunto de pruebas (2do de la segunda etapa de sintonización): evaluación de reglas y tasas de aprendizaje, para un tamaño de población de 48 soluciones

En el segundo conjunto de pruebas, donde el tamaño de la población corresponde a 48 soluciones, se empleó el mismo criterio de eliminación para seleccionar las alternativas que sobrevivirían para las pruebas siguientes, en las cuales la población será reducida a 32 soluciones. La lista de alternativas sobrevivientes es igual a la mencionada en el caso anterior, con la diferencia de que, algunas pasaron con un menor número de datos por debajo del $MSE = 0.0225$ con respecto a su condición anterior; en otras palabras, a pesar que el requisito se cumple la calidad sufrió una pequeña degradación.

- Cuarto y quinto conjunto de pruebas (3ro y 4to de la segunda etapa del proceso de sintonización): evaluación de reglas y tasas de aprendizaje, para los tamaños de población de 16 y 32 soluciones

Dada la constancia del número de soluciones sobrevivientes y que la variación en calidad no es significativa, se decide restringir aún más el requisito de eliminación: para las poblaciones de 32 y 16 soluciones se evaluaron sólo aquellas alternativas que para el tamaño de población anteriormente evaluado se cumple que el 100% de sus datos está por debajo del $MSE = 0.0225$. En ambos casos, las listas de alternativas sobrevivientes corresponden a las reglas de aprendizaje exponencial y sigmoide en las que se involucra la tasa de aprendizaje cero.

- Análisis gráfico de los resultados

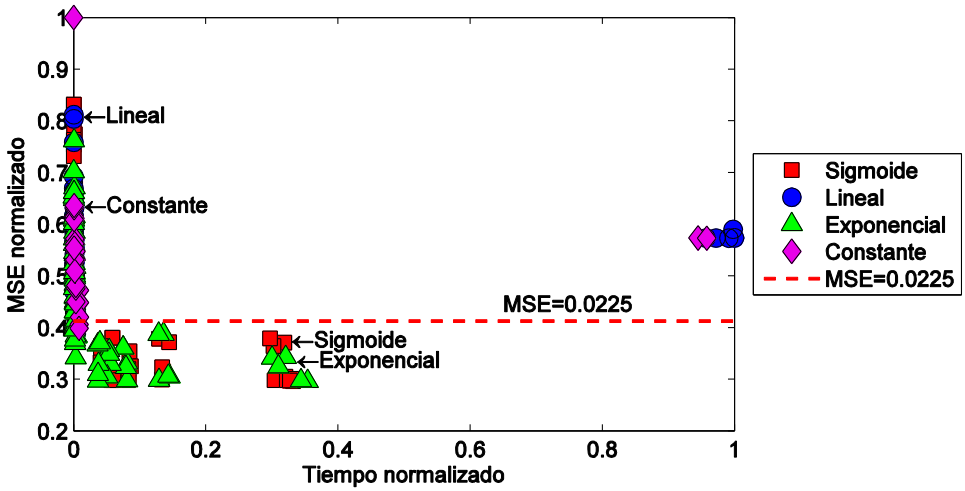
En total se evaluaron 91 combinaciones, cuando en un principio, incluyendo las variaciones hechas al tamaño de la población, se disponía de 204. Demostrando que el curso seguido para las pruebas de sintonización redujo sustancialmente las alternativas a evaluar. De las 91 se analizaron sólo aquellas que 100% de sus resultados se encuentran por debajo del $MSE = 0.0225$. Bajo éste criterio el número de alternativas se reduce a 40 y las listas de alternativas provenientes de las pruebas con tamaño de población de 64 y 48 soluciones quedan iguales a las obtenidas para 32 y 16.

En principio, los resultados son separados y analizados por tamaño de población y tipo de regla de aprendizaje (ver desde la Figura 2-35 a la Figura 2-42). De entre cada grupo se seleccionaron las mejores alternativas, para las cuales se realizaron nuevas pruebas con el fin de reforzar la información obtenida y sus resultados fueron reagrupados según regla de aprendizaje (ver Figura 2-43 y Figura 2-44). Posteriormente, las alternativas destacadas de estos subconjuntos fueron reunidas para el análisis final (ver Figura 2-45).
En la Figura 2-31 (ver también Figura 2-32) se presentan los resultados de las 51 combinaciones posibles entre reglas y tasas de aprendizaje, para el tamaño de población de 64 soluciones. Se muestra el comportamiento general de las reglas de aprendizaje contempladas y se observa que existe una similitud en entre ellas, debido a que obedecen a una lógica en común, ya mencionada: En general, el uso de tasas pequeñas favorece la exploración del espacio de soluciones, a costa de tiempos de convergía extensos (a lo cual corresponden los datos ubicados abajo y a la derecha del cuadrante). Por otra parte, el uso de valores grandes, tendientes a la unidad, propende por acelerar el tiempo de convergencia por medio de la explotación de las soluciones, a riesgo de precipitar la entrega de resultados, que en consecuencia, tienden a ser de mala calidad (a lo cual corresponden los datos ubicados hacía arriba y a la izquierda del cuadrante). Los datos en medio de ambas condiciones, próximos al origen, emplean combinaciones que procuran un equilibrio entre exploración y explotación, o visto de otra manera, entre calidad y tiempo.

Adicionalmente, en la Figura 2-33 (ver también la Figura 2-34) se muestra individualizado el comportamiento de la regla sigmoide y se resaltan las diferentes combinaciones de tasas de aprendizaje, permitiendo observar que en general, las variaciones de calidad de las aproximaciones obedecen particularmente a las variaciones de la regla de aprendizaje mínima y las variaciones temporales a las variaciones de la regla de aprendizaje máxima, de la siguiente manera: La calidad

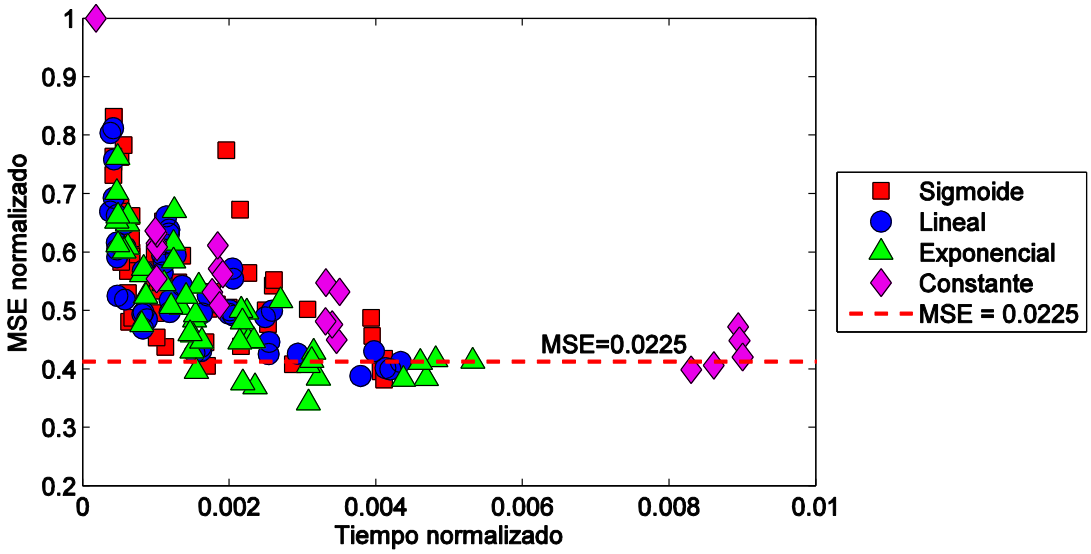
aumenta conforme se reduce la tasa de aprendizaje mínima y, por otro lado, el tiempo de convergencia disminuye si la tasa de aprendizaje máxima aumenta. Así, calidad y tiempo, mantienen una relación inversa con LR_{min} y LR_{max}, respectivamente.

Figura 2-31: Sintonización APBIL: comportamiento general de las reglas de aprendizaje frente a variaciones de las tasas de aprendizaje, para un tamaño de población de 64 soluciones.



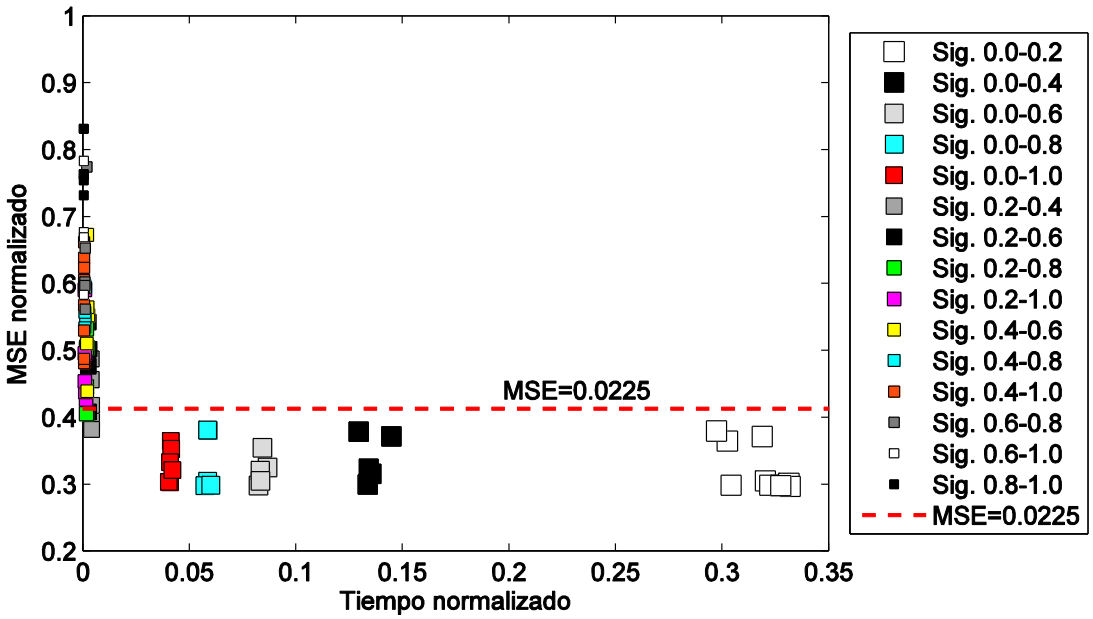
Nombre de la fuente: Elaboración propia

Figura 2-32: Sintonización APBIL: comportamiento general de las reglas de aprendizaje frente a variaciones de las tasas de aprendizaje, para un tamaño de población de 64 soluciones, vista del rango temporal [0, 0.001].



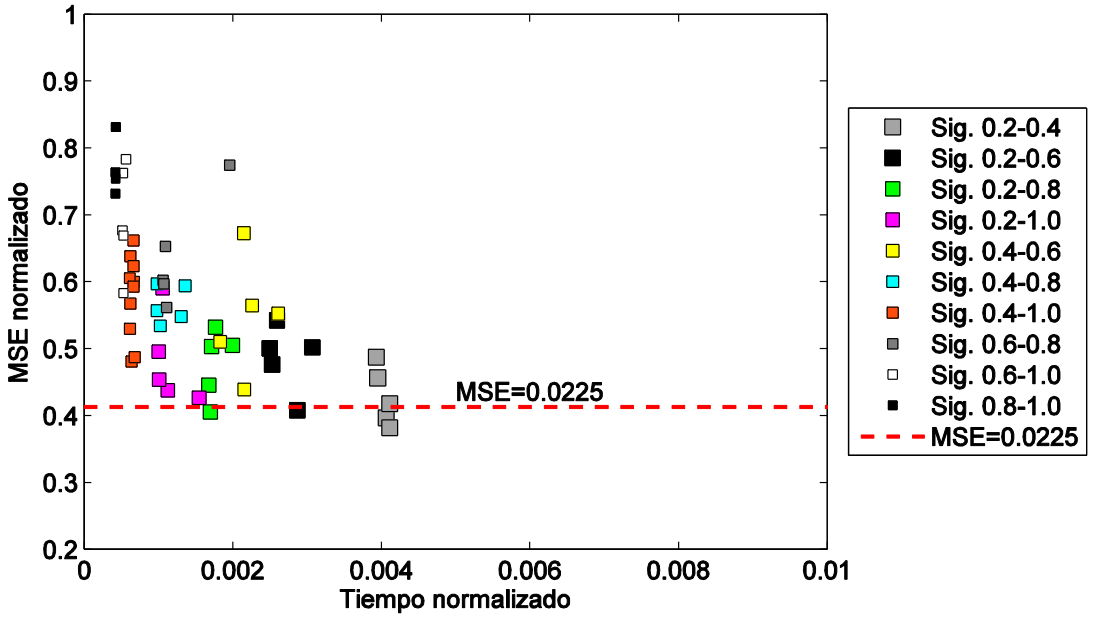
Nombre de la fuente: Elaboración propia

Figura 2-33: Sintonización APBIL: comportamiento de la regla de aprendizaje sigmoide frente a variaciones de las tasas de aprendizaje.



Nombre de la fuente: Elaboración propia

Figura 2-34: Sintonización APBIL: comportamiento de la regla de aprendizaje sigmoide frente a variaciones de las tasas de aprendizaje, vista del rango temporal [0, 0.001].



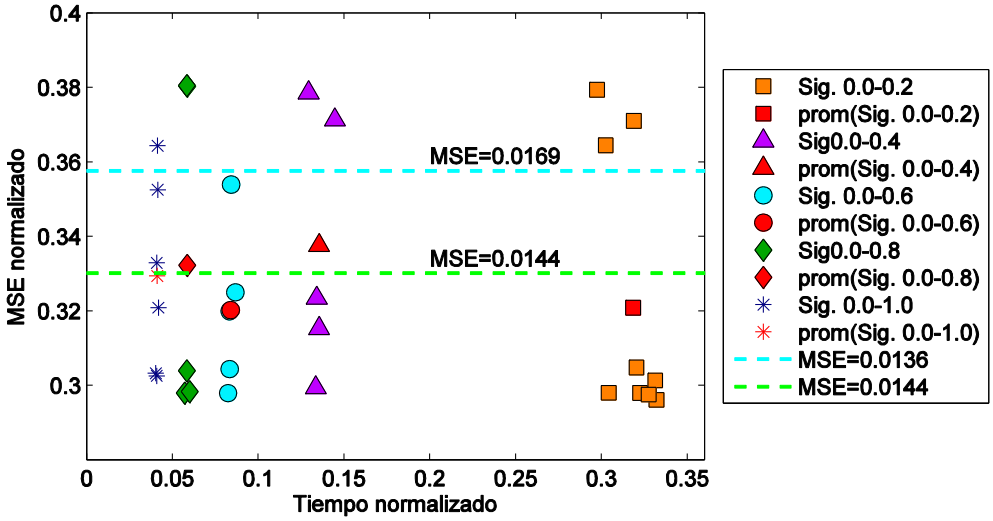
Nombre de la fuente: Elaboración propia

Ahora bien, en la Figura 2-35 y en la Figura 2-36 se presentan los resultados de las alternativas consideradas por el tamaño de población de 64 soluciones, discriminadas por regla de aprendizaje. Éstas corresponden a las reglas exponencial y sigmoide en las que participa la tasa de aprendizaje cero. En la imagen se puede observar que los promedios de las series rondan el $MSE = 0.0144$ y en general, conservan una distribución similar respecto al eje vertical; permitiendo afirmar que, en cuanto a calidad, son todas opciones aptas. Entre ellas se ha elegido las mejores alternativas por regla de aprendizaje:

- Para la regla sigmoide se elige como mejor alternativa la pareja de tasa de aprendizaje (0.0, 0.6), dado que está en el grupo de las más rápidas, su MSE promedio es el mejor y es la única cuyo máximo no supera el $MSE = 0.0169$, demostrando mayor estabilidad.

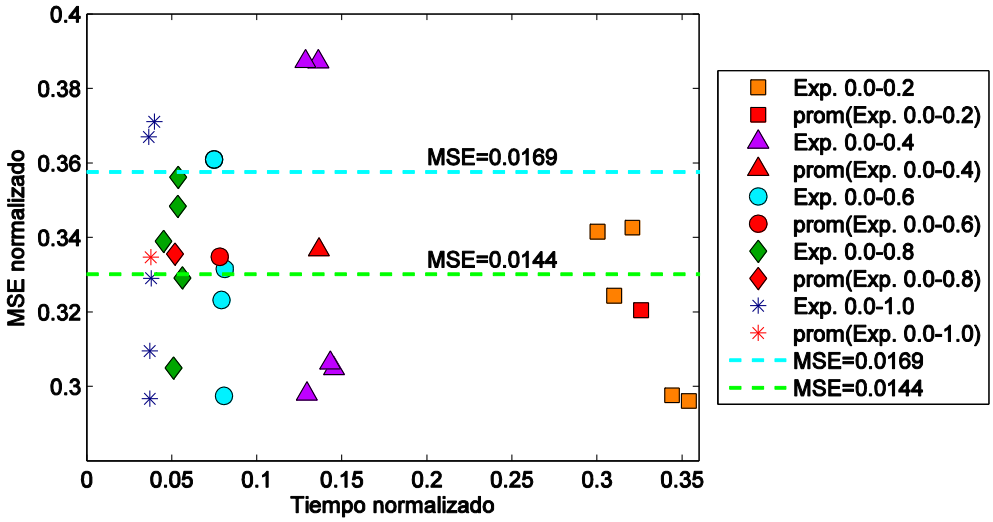
- Para la regla exponencial la pareja elegida es (0.0, 0.8), ya que es la segunda más veloz y a diferencia de las demás que también brindan una respuesta rápida, es la única cuyo máximo no supera el $MSE = 0.23$.

Figura 2-35: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 64 soluciones y regla de aprendizaje sigmoide.



Nombre de la fuente: Elaboración propia

Figura 2-36: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 64 soluciones y regla de aprendizaje exponencial.



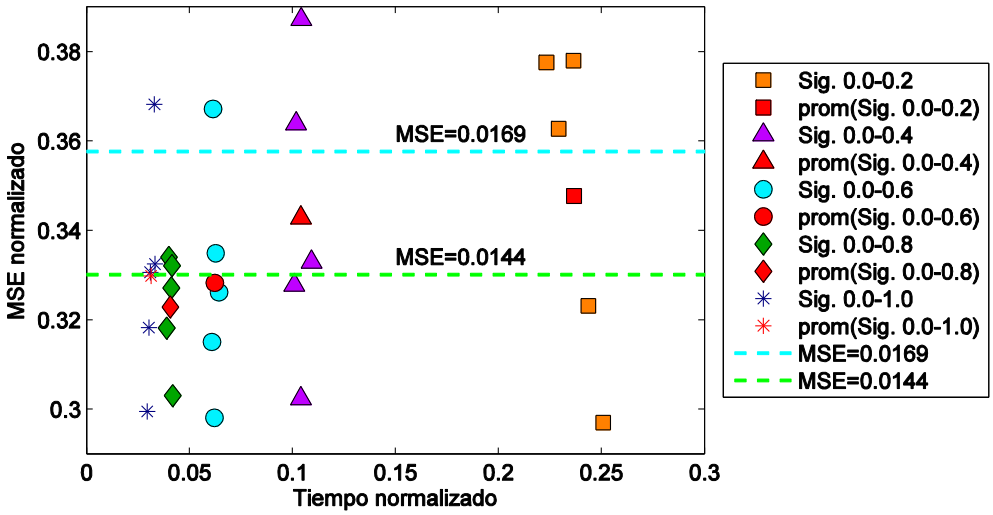
Nombre de la fuente: Elaboración propia

En las Figuras 2-37 a 2-40 se presentan las alternativas consideradas por el tamaño de población de 48 y 32 soluciones, discriminadas por el tipo de regla de aprendizaje. Las soluciones escogidas como las mejores alternativas son iguales en ambos casos:

- Para la regla sigmoide, en ambas situaciones, la pareja (0.0, 0.6) se presenta como la segunda alternativa más rápida, la de mejor MSE promedio y cuyo MSE máximo es considerablemente más bajo que el máximo de las demás alternativas.

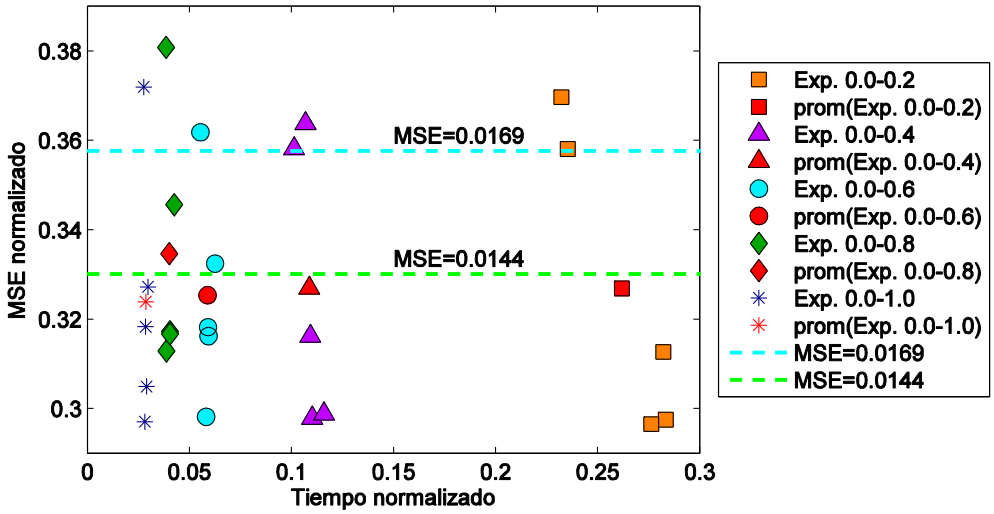
- En cuanto a la regla exponencial, en ambos casos, la pareja (0.0, 1.0) es la opción más rápida y con el mejor promedio.

Figura 2-37: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 48 soluciones y regla de aprendizaje sigmoide.



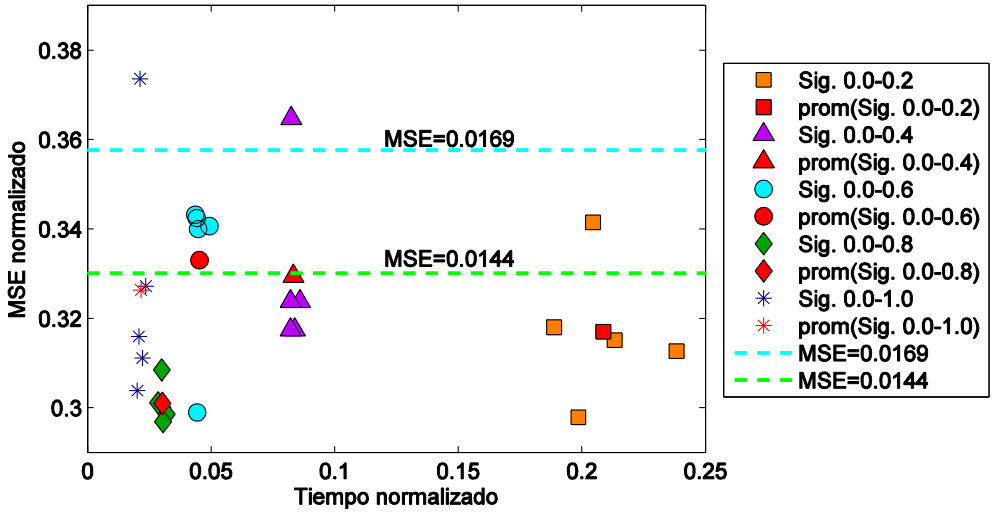
Nombre de la fuente: Elaboración propia

Figura 2-38: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 48 soluciones y regla de aprendizaje exponencial.



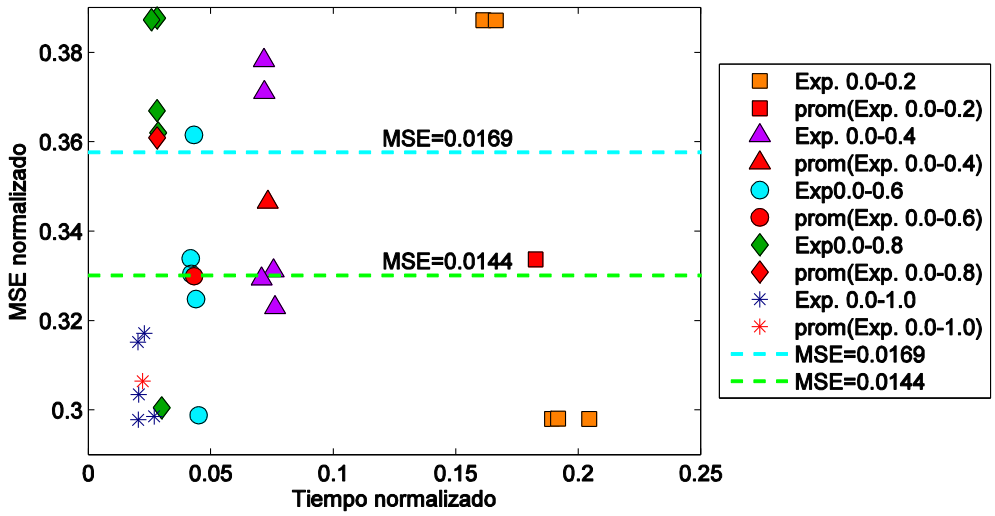
Nombre de la fuente: Elaboración propia

Figura 2-39: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 32 soluciones y regla de aprendizaje sigmoide.



Nombre de la fuente: Elaboración propia

Figura 2-40: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 32 soluciones y regla de aprendizaje exponencial.



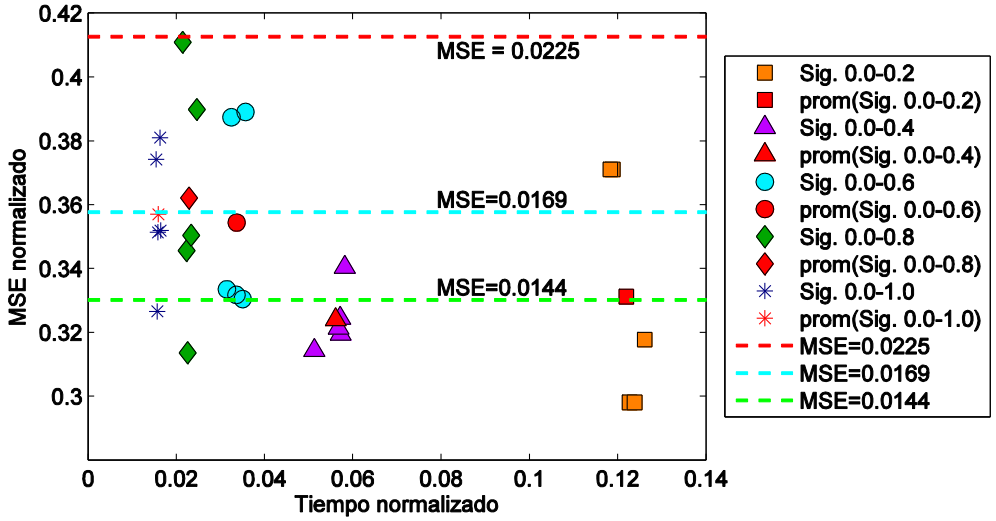
Nombre de la fuente: Elaboración propia

Por último, en la Figura 2-41 y en la Figura 2-42 se presentan las alternativas consideradas por el tamaño de población de 16 soluciones, discriminadas por el tipo de regla de aprendizaje. A continuación se presentan las alternativas consideradas como las mejores, que, aunque no son las más veloces de sus respectivos grupos, sus tiempos de convergencia son comparables con los tiempos de las mejores alternativas pertenecientes a tamaños de población más grandes:

- Para la regla sigmoide la pareja elegida es (0.0, 0.4), porque, aunque no es la más rápida, es la primera cuyo promedio está alrededor del $MSE = 0.0144$. Mientras que las demás, poseen promedios que rondan el $MSE = 0.0169$.

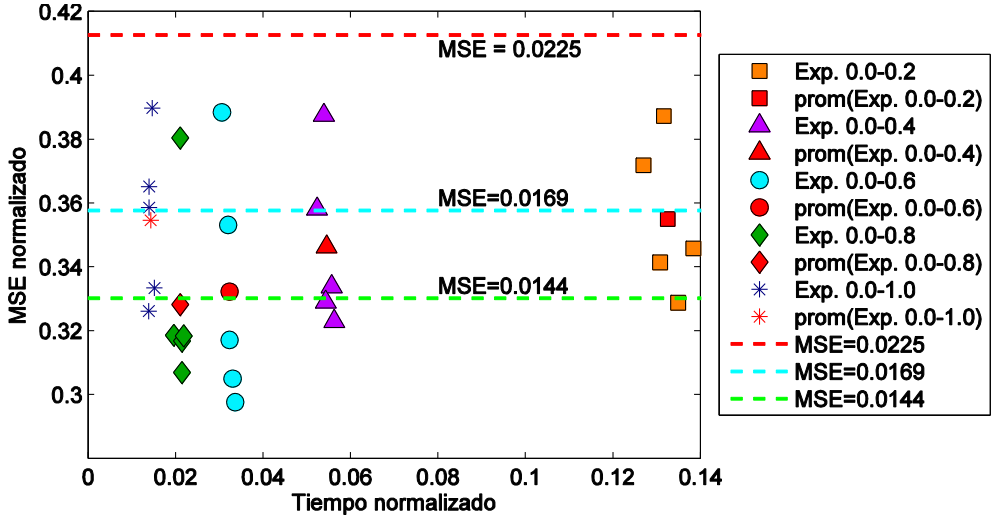
- En cuanto a la regla exponencial, la pareja (0.0, 0.8) se presenta como al asegunda más veloz y cuyo MSE promedio es el más bajo.

Figura 2-41: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 16 soluciones y regla de aprendizaje sigmoide



Nombre de la fuente: Elaboración propia

Figura 2-42: Sintonización APBIL: alternativas para la sintonización, consideradas por el tamaño de población de 16 soluciones y regla de aprendizaje exponencial.



Nombre de la fuente: Elaboración propia

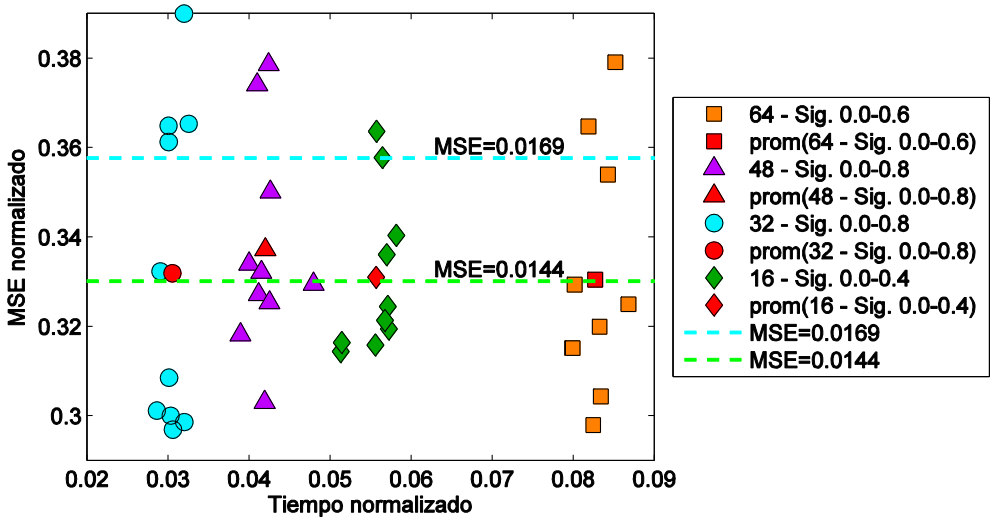
En la Figura 2-43 y en la Figura 2-44 se reúnen los resultados de las alternativas destacadas por cada tamaño de población, discriminadas por el tipo de regla de aprendizaje. Exceptuando a la regla exponencial por el tamaño de población de 16 soluciones, el promedio de las alternativas se aproximan al $MSE = 0.0144$, demostrando competitividad. De entre ellas, las elegidas como las mejores son:

- Para la regla sigmoide se elige el tamaño de población de 32 soluciones, cuya pareja de tasas de aprendizaje es (0.0, 0.8), a causa de que es la más veloz.

- Para la regla exponencial se elige el tamaño de población de 32 soluciones, cuya pareja de tasas de aprendizaje es (0.0, 1.0), dado que su MSE promedio es el mejor y es la segunda alternativa más rápida, después de la de 16 soluciones.

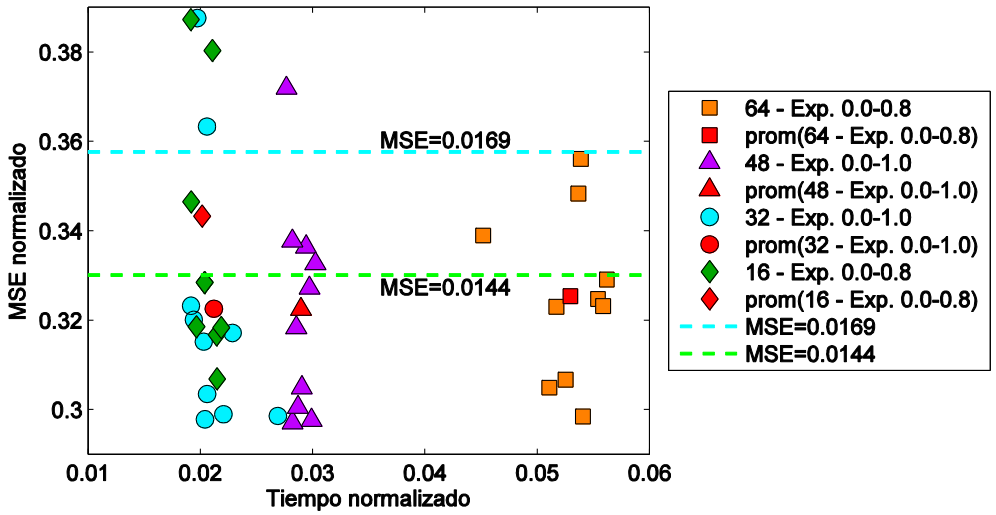
De lo anterior, se tiene que el tamaño elegido para la población es de 32 soluciones. Debido a que las mejores alternativas para la sintonización del algoritmo coinciden en la elección del tamaño de población, que además fueron evaluados otros tamaños por encima y por debajo del elegido y dado el exhaustivo proceso de búsqueda llevado acabo para determinar los valores más convenientes para los parámetros del algoritmo, se concluye que: cuando se trata del algoritmo APBIL dedicado al diseño de filtros digitales lineales e invariantes en el tiempo, el tamaño óptimo para la población es igual, o bien está próximo, a 32 soluciones.

Figura 2-43: Sintonización APBIL: mejores alternativas para la sintonización, por tamaño de población, para la regla de aprendizaje sigmoide.



Nombre de la fuente: Elaboración propia

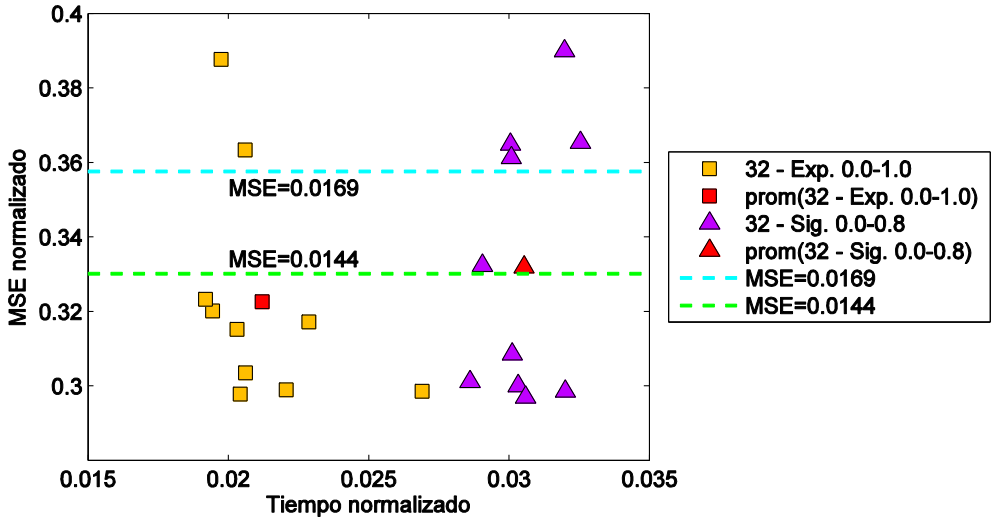
Figura 2-44: Sintonización APBIL: mejores alternativas para la sintonización, por tamaño de población, para la regla de aprendizaje exponencial.



Nombre de la fuente: Elaboración propia

Las dos alternativas resultantes en definitiva son aptas y recomendables para la sintonización del algoritmo ABPIL, sin embargo, hay que determinar cuál de las dos es la mejor opción de entre todas las vistas hasta ahora. En la Figura 2-45 se contrastan sus resultados: La diferencia temporal es poca, y sus distribuciones respecto al eje vertical son similar, e incluso sus máximos y mínimos se encuentran en posiciones semejantes. La pequeña diferencia en calidad hace que el MSE promedio de la regla exponencial se mejor. Además, es ésta la más rápida de las dos. Por tanto, se elige la regla de aprendizaje exponencial, con tasas de aprendizaje (0.0, 1.0) para sintonizar el algoritmo APBIL.

Figura 2-45: Mejores alternativas para la sintonización del algoritmo APBIL.



Nombre de la fuente: Elaboración propia

- Sintonización recomendada para el algoritmo APBIL

Finalmente, se ha encontrado la sintonización idónea para el algoritmo APBIL. El resumen de la sintonización recomendada se presenta en la Tabla 2-15. Adicionalmente, en las Tablas 2-13 y 2-14 se proporcionan ajustes alternativos, que favorecen el desempeño del algoritmo bajo condiciones de operación diferentes a las contempladas por la sintonización sugerida. Valores o combinaciones diferentes a las recomendadas con garantizan el rendimiento óptimo del algoritmo, ni el hallazgo de buenas soluciones.

Tabla 2-13: Sintonización alternativa para el algoritmo APBIL.

Parámetro	Sintonización recomendada
Representación binaria ($Optbr$)	*Punto fijo – complemento a dos*
Tamaño de la población ($popsize$)	32 soluciones
Regla de aprendizaje	Sigmoide
Tasa de aprendizaje mínima (LR_{min})	1×10^{-7}
Tasa de aprendizaje máxima (LR_{max})	0.8
Criterio elitista	Activo
Número de soluciones élite	1 solución
Criterios de parada	
Tolerancia de la entropía ($tolE$)	1×10^{-9}
Criterio de parada por alcance del límite de iteraciones	Activo
Límite de iteraciones ($iterlimit$)	20000 iteraciones

Tabla 2-14: Sintonización alternativa para el algoritmo APBIL: reglas y tasas de aprendizaje para diferentes tamaños de población.

Tamaño de la población ($popsize$)	Regla de aprendizaje	Tasa de aprendizaje mínima (LR_{min})	Tasa de aprendizaje máxima (LR_{max})
(0, 24]	Sigmoide	0.0	0.4
(24, 56]	Exponencial	0.0	1.0
(56, Inf)	Exponencial	0.0	0.8

Tabla 2-15: Sintonización recomendada para el algoritmo APBIL.

Parámetro	Sintonización recomendada
Representación binaria ($Optbr$)	*Punto fijo – complemento a dos*
Tamaño de la población ($popsize$)	32 soluciones
Regla de aprendizaje	Exponencial
Tasa de aprendizaje mínima (LR_{min})	1×10^{-7}
Tasa de aprendizaje máxima (LR_{max})	1.0
Criterio elitista	Activo
Número de soluciones élite	1 solución
Criterios de parada	
Tolerancia de la entropía ($tolE$)	1×10^{-9}
Criterio de parada por alcance del límite de iteraciones	Activo
Límite de iteraciones ($iterlimit$)	20000 iteraciones

2.5 Aplicativo de Diseño de Filtros Digitales basado en las herramientas de optimización APBIL, GA y TS

Como resultado de la implementación de los algoritmos APBIL, GA y TS para el diseño de filtros digitales lineales e invariantes en el tiempo, se construyó una interfaz de usuario, amigable, que permite realizar diseños FIR e IIR empleando cualquiera de las herramientas de optimización metaheurística propuestas. Dicho *software*, presenta las mismas ventajas que las metodologías de diseño presentadas, pues a diferencia de muchas otras alternativas de *software* de diseño de filtros digitales, contempla la cuantificación y codificación desde un comienzo del proceso de diseño, y no solo al final del mismo. Adicionalmente, permite obtener el diseño bajo las técnicas tradicionales más conocidas, proporcionando el valor de los coeficientes ideales (previos a la cuantificación) y cuantificados. Además, permite realizar comparaciones gráficas de los Espectros de Frecuencia y de los errores de los diseños obtenidos con las herramientas de optimización metaheurística, entre sí y contra los obtenidos por las técnicas tradicionales.

El *software* brinda acceso a los parámetros de los algoritmos y contempla como valores por defecto los obtenidos por los procesos de sintonización (ver las Tablas 2-9, 2-11 y 2-15). Igualmente,

brinda acceso a los parámetros requeridos por las técnicas tradicionales de diseño, para las cuales se emplearon las rutinas disponibles en Matlab R2013a, para Windows, y sus valores por defecto son los sugeridos por éstas.

Aunque el *software* originalmente fue construido como una aplicación GUIDE de Matlab, la versión final es una aplicación independiente para Windows, que permite imprimir y almacenar los diseños generados en un archivo script, que se genera dentro de la misma dirección o carpeta en la cual se encuentra el aplicativo. Internamente, el *software* contiene todas las rutinas propias de Matlab que se utilizaron en la GUIDE. Las técnicas tradiciones contempladas y las ventanas empleadas por los métodos de Enventanado y Muestreo en Frecuencia (implementadas a través de rutinas de Matlab) se listan en la Tabla 2-16 y en la Tabla 2-17, respectivamente.

Tabla 2-16: Técnicas tradicionales de diseño contempladas.

Respuesta impulsiva	Técnica tradicional de diseño
IIR	Filtro digital Notch
	Filtro digital Peak
	Filtro digital Butterworth
	Filtro digital Chebyshev tipo I
	Filtro digital Chebyshev tipo II
	Filtro digital Elíptico o de Causer
	Filtro digital Yulewalk (Método de Mínimos Cuadrados)
FIR	Método de Enventanado (fase lineal)
	Método de Muestreo en Frecuencia
	Método de Mínimos Cuadrados (fase lineal)
	Algoritmo de Parks-McClellan

Tabla 2-17: Ventas contempladas para los Métodos de Enventanado y Muestreo en Frecuencia.

No.	Nombre de la ventana
1	Ventana Bartlett
2	Ventana Bartlett-Hann
3	Ventana Blackman
4	Ventana Blackman-Harris
5	Ventana Bohman
6	Ventana Chebyshev
7	Ventana de Superficie Plana
8	Ventana Gaussiana
9	Ventana Hamming
10	Ventana Hann
11	Ventana Kaiser
12	Ventana Nuttall's Balckman-Harris
13	Ventana Parzen (de la Valle-Poussin)
14	Ventana Rectangular
15	Ventana Triangular

Además, el software cuenta con dos elementos extra, que pueden ser de ayuda para obtener mejores diseños: insertar la solución anterior como semilla en un nuevo proceso de diseño y la comparación no uniforme del MSE.

- Es posible emplear la solución anteriormente obtenida, siempre y cuando cumpla las condiciones requeridas, para generar a partir de ella un nuevo diseño. Las condiciones son: que exista una solución anterior, y que entre la nueva y la anterior solución coincidan en el tamaño de la representación ($v \times k$ bits) y el tipo de simetría, si aplica. En el GA la semilla entra a hacer parte de la primera población y muy seguramente del grupo élite, en el algoritmo TS ingresa como la solución inicial, y en algoritmo APBIL afecta en el inicio al arreglo de probabilidades, incrementando la probabilidad de ocurrencia de la semilla en un porcentaje predefinido.

  La utilización de estas semillas permite actuar a los algoritmos como si estuviesen conectados en cascada para producir en conjunto una solución. Por ejemplo, en la Tabla 2-18 se muestran los errores obtenidos con una sola ejecución de los algoritmos, sin la

semilla, y con la posterior intervención del mismo u otro algoritmo, empleando la anterior solución como semilla.

Tabla 2-18: Utilización de la solución anterior como semilla.

1er diseño		2do diseño (empleando semilla)		Características del diseño
Algoritmo	MSE	Algoritmo	MSE	
APBIL	0.009845	APBIL (60%)[4]	0.008151	IIR, LP, orden 9 (20 *tabs*), 32 bits por *tab*, punto fijo
	0.009586	GA	0.008328	
	0.009743	TS	0.009360	
GA	0.014009	GA	0.013731	FIR, HP, simetría irregular, orden 16, 64 bits por *tab*, punto fijo
	0.014132	APBIL (60%)[4]	0.013070	
	0.012229	TS	0.011932	
TS	0.024415	TS	0.023508	IIR, PB, orden 6 (14 *tabs*), 8 bits por *tab*, punto fijo
	0.026869	APBIL (60%)[4]	0.022323	
	0.019881	GA	0.018568	

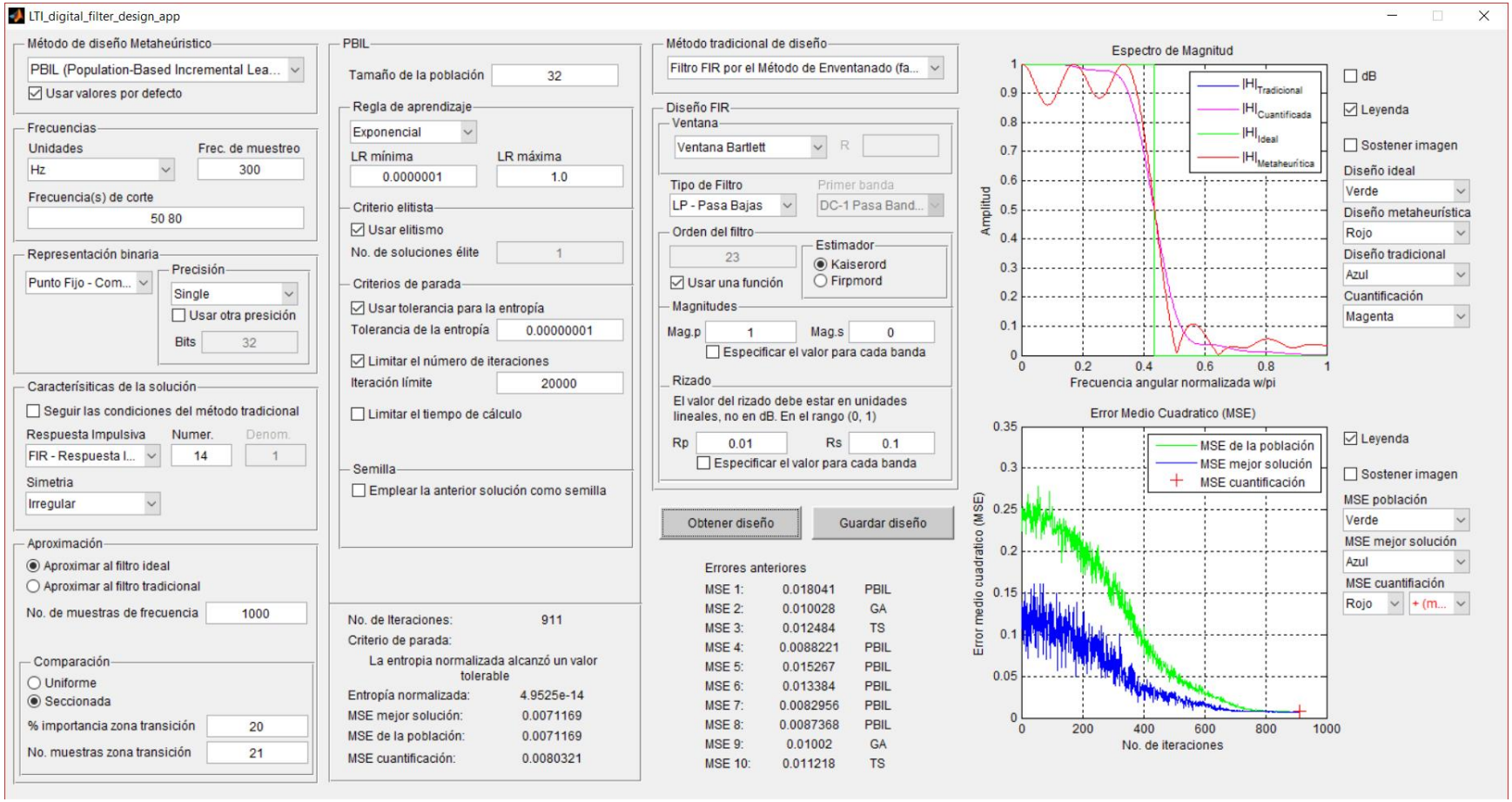
[4]Porcentaje de afectación de la semilla sobre el arreglo de probabilidades

- Por otra parte, la comparación no uniforme del MSE permite asignar un peso (w) o porcentaje de importancia a la(s) zona(s) de transición. Con lo cual, el aporte a la evaluación del MSE de las muestras que hacen parte de la(s) zona(s) de transición es afectado por dicho factor, mientas que para las que no lo son, es afectado por el complemento, como se muestra en la Ecuación (2-15).

$$MSE = w \times \sum MSE_{transiton\ zone} + (1 - w) \times \sum MSE_{others} \qquad (2\text{-}15)$$

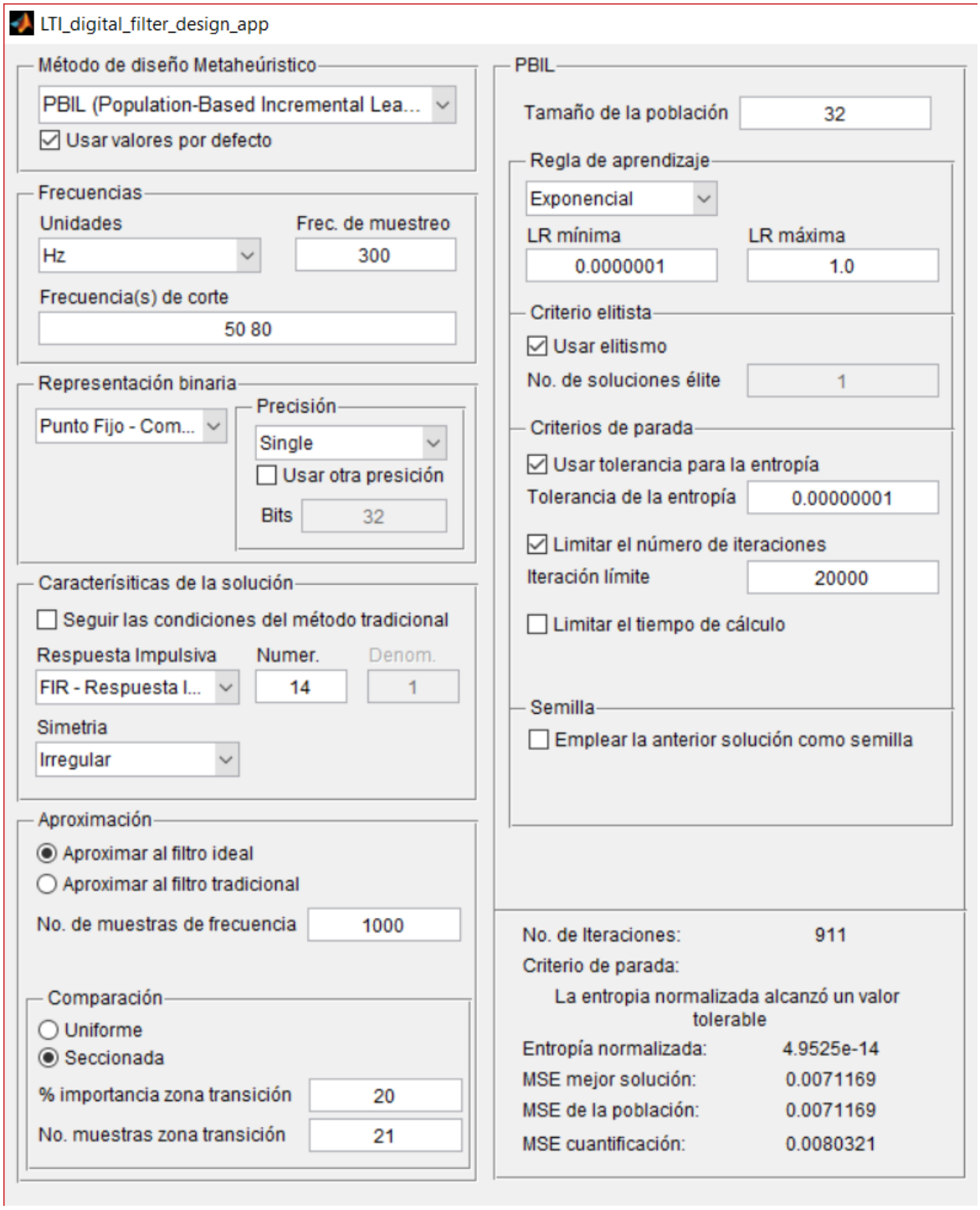
Finalmente, en la Figura 2-46 (ver también Figura 2-47 y Figura 2-48) se muestra una imagen de la interfaz de usuario. En la cual se pude apreciar los botones de comando empleados para un diseño FIR, bajo la herramienta de optimización APBIL, contrastado con un diseño obtenido por medio del Método de Enventanado. A la derecha de la imagen se observan los Espectros de Frecuencia del diseño APBIL (rojo), del diseño tradicional sin cuantificar (azul) y cuantificado (magenta), y del filtro deseado, un filtro ideal (verde), el cual sirve de referencia para el cálculo del MSE, cuya evolución también se muestra en la figura.

Figura 2-46: Interfaz de usuario del Aplicativo de diseño de Filtros Digitales.



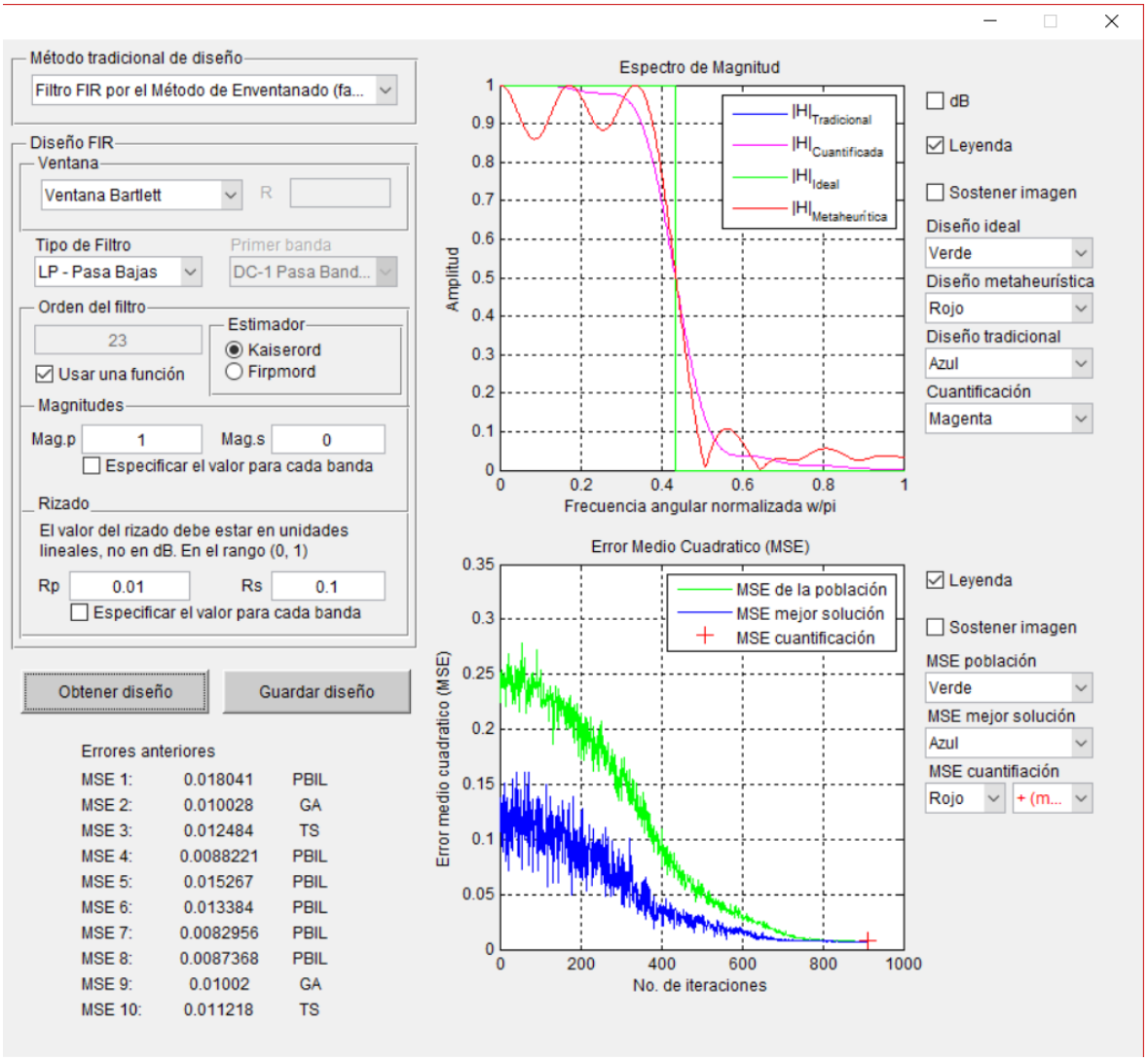
Nombre de la fuente: Elaboración propia

Figura 2-47: Interfaz de usuario del Aplicativo de diseño de Filtros Digitales: vista de los botones de comando para realizar el diseño a través de la herramienta metaheurística APBIL.



Nombre de la fuente: Elaboración propia

Figura 2-48: Interfaz de usuario del Aplicativo de diseño de Filtros Digitales: vista de los botones de comando para el diseño a través del Método de Enventanado y de las gráficas de Espectro de Frecuencia y MSE.



Nombre de la fuente: Elaboración propia

3. Resultados

En el capítulo actual se presentan algunos de los resultados obtenidos con el Aplicativo de Diseño de Filtros Digitales lineales e invariantes en el tiempo, basado en las herramientas de optimización metaheurística APBIL, GA y TS. Su implementación fue tema de discusión del capítulo 2, la presentación de cada uno de los algoritmos mencionados, dentro del diseño de filtros digitales, se presentó en las secciones 2.1 a 2.3, y los procesos de sintonización de las variables de control (o parámetros) de cada algoritmo en la sección 2.4.

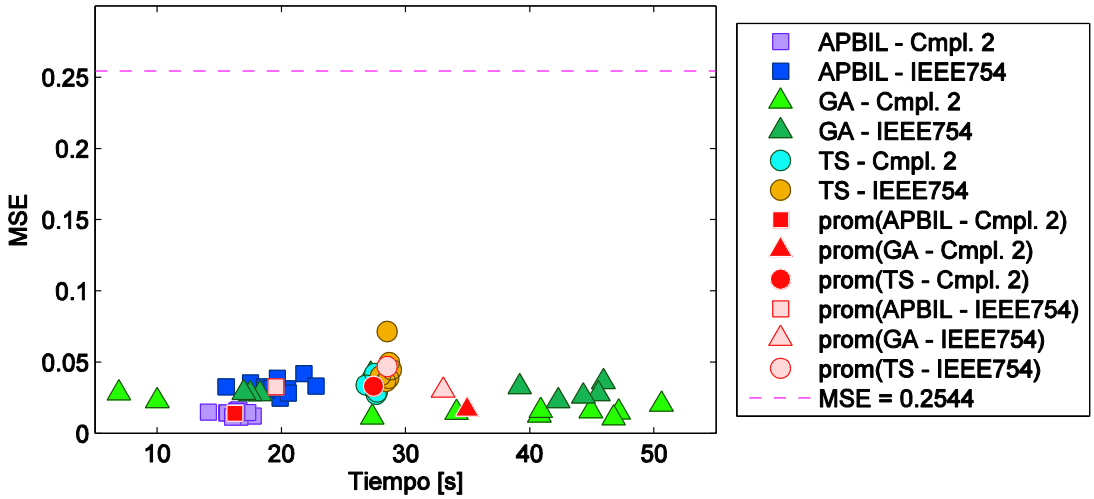
En el siguiente conjunto de gráficas se muestra el desempeño de las herramientas de optimización mencionadas, en el diseño de filtros digitales FIR e IIR, en contraste con los métodos de diseño tradicionales. Los errores fueron medidos tomando como referencia el diseño ideal, según la Ecuación (2-4), Ecuación del MSE. Con el fin de realizar una comparación justa, los diseños tradicionales y los obtenidos por las herramientas de optimización metaheurística tienen el mismo número de coeficientes (o *tabs*), y su cuantificación se realizó bajo las mismas precisión y representación numérica. En cada caso se reportan los resultados para el uso de las representaciones de punto fijo (complemento a dos) y punto flotante (según el estándar IEEE 754), con precisiones de 32 y 64 bits por *tab*.

- Comparación entre de diseños IIR

En la Figura 3-1 se muestran los errores y los tiempos de convergencia obtenidos en el diseño de un filtro IIR, LP, de cuarto orden (10 *tabs*), comparado con un diseño Chebishev tipo II, para una precisión de $k = 32$ bits por *tab*. La línea horizontal representa el $MSE = 0.2544$ perteneciente al filtro obtenido bajo el método tradicional (aunque la representación numérica varía, para $k = 32$ bits, los errores de la cuantificación son aproximadamente iguales). Los diferentes puntos de una misma serie de datos corresponden a diferentes ejecuciones del mismo algoritmo bajo la misma parametrización que, como se explicó en el capítulo anterior, no siempre proporciona exactamente el mismo resultado.

Se evidencia que los tres métodos de diseño son competentes en calidad, entre sí y respecto al diseño tradicional. En este caso particular, superan ampliamente el MSE del diseño Chebyshev tipo I. En cuanto a los tiempos de convergencia, aunque existen diferencias, ninguno superó el minuto.

Figura 3-1: Prueba de desempeño: diseño Chebyshev tipo II, LP, de cuarto orden.

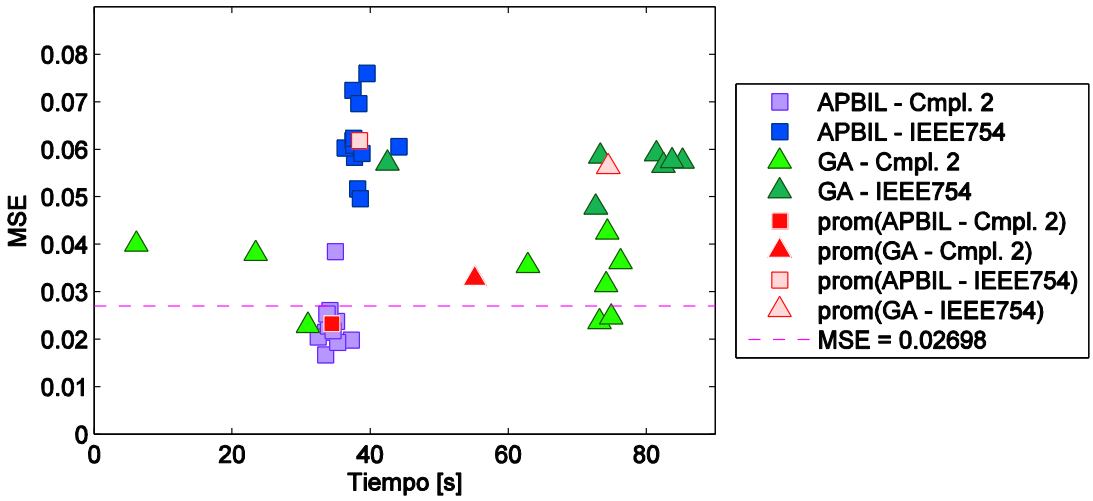


Nombre de la fuente: Elaboración propia

En la Figura 3-2 se muestran los errores y los tiempos de convergencia para un diseño IIR, SB, de quinto orden (22 *tabs*), contrastado frente a un diseño Elíptico o de Causer, para una precisión de $k = 64$ bits por *tab*. En éste caso, los resultados proporcionados por el algoritmo TS fueron de mala calidad y desproporcionales en tiempo (por tal razón, se decidió excluirlos de la gráfica, para ganar resolución en la observación de las demás series de datos).

Todas las alternativas presentadas ofrecen resultados de calidad, pero solo aquella que implementa el algoritmo APBIL y la representación binaria *punto fijo – complemento a dos* brinda soluciones que superan al diseño tradicional ($MSE = 0.02698$), con errores que se sitúan alrededor del $MSE = 0.02324$. También es notable que, las alternativas que vinculan a la representación de *punto flotante – IEEE 754* en ninguna ocasión alcanzó a superar al diseño tradicional.

Figura 3-2: Prueba de desempeño: diseño Elíptico o de Causer, SB, de orden 22.



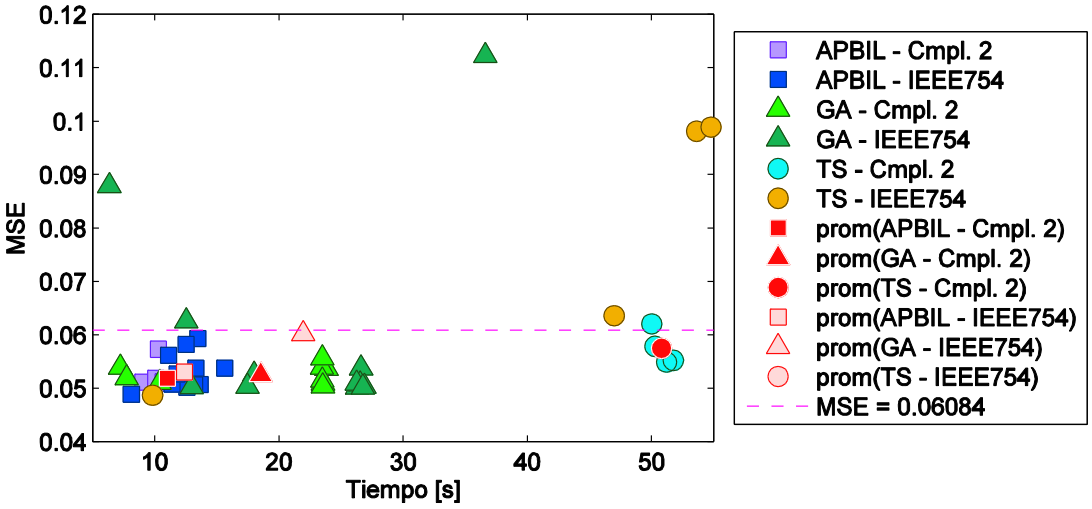
Nombre de la fuente: Elaboración propia

- Comparación entre diseños FIR

En la Figura 3-3 y en la Figura 3-4 se compara el desempeño de los algoritmos propuestos con respecto a dos de los métodos de optimización clásica comúnmente empleados en el diseño fe filtros FIR. En la Figura 3-3 se expone el caso de un filtro PB, de séptimo orden, contrastado con el diseño (simétrico y de fase lineal) obtenido por medio del método de Mínimos Cuadrados, para una precisión de $k = 64$ bits por *tab*. Por otro lado, en la Figura 3-4, se muestran los resultados para un filtro HP, de noveno orden, confrontado con un diseño (anti simétrico) obtenido bajo el algoritmo Parcks-McClellan, para una precisión de $k = 32$ bits por *tab*.

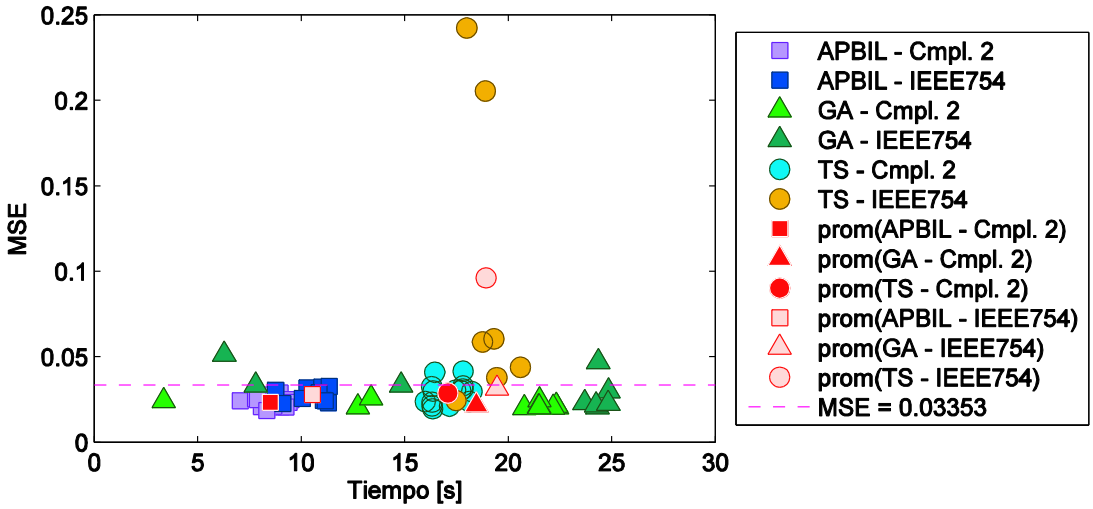
En los casos expuestos, los algoritmos de optimización metaheurística demostraron estar en capacidad de proporcionar resultados de mayor calidad que las metodologías tradicionales, basadas en optimización clásica. Tan solo la alternativa que implementa a la vez el algoritmo TS y la representación de *punto flotante – IEEE 754* presentó resultados poco competentes (visible en el caso del filtro PB, de séptimo orden) y de mala calidad (en el caso del filtro HP, de noveno orden). Con el fin de conservar resolución, en la siguiente gráfica no se muestran algunos puntos de dicha serie de datos.

Figura 3-3: Prueba de desempeño: diseño FIR, por el método de Mínimos Cuadrados, PB, de séptimo orden.



Nombre de la fuente: Elaboración propia

Figura 3-4: Prueba de desempeño: diseño FIR, por el algoritmo Parks-McClellan, HP, de noveno orden.



Nombre de la fuente: Elaboración propia

- Resumen de las pruebas de desempeño

En general, la competitividad respecto a los diseños tradicionales, cuantificados, se reduce conforme aumenta el orden del filtro, es decir, entre más *tabs* se usen para representarlo, al igual

que si la precisión binaria aumenta. Lo cual se debe a dos factores: En primer lugar, aumentar la precisión en bits, y en especial el orden del filtro, mejora la aproximación del diseño tradicional, al filtro ideal. Por otro lado, para los algoritmos optimización metaheurística, aumentar el número de bits de la representación de la solución del filtro hace más complejo el proceso de búsqueda del diseño óptimo, pues cada bit adicional representa una nueva variable que se suma al problema, aumentando su grado de dificultad. Sin embargo, las metodologías de diseño propuestas demostraron ser adecuadas para el diseño de filtros FIR e IIR, especialmente los algoritmos APBIL y GA.

En las Figuras 3-5 y 3-6 se reúnen un aglomerado de datos, resultantes de la evaluación de diferentes condiciones (es decir, de distintos diseños). Aunque en éstas sólo se hace la distinción entre los tipos de representación numérica y los algoritmos optimización metaheurística implementados, permiten observar tendencias que caracterizan las respuestas de cada algoritmo:

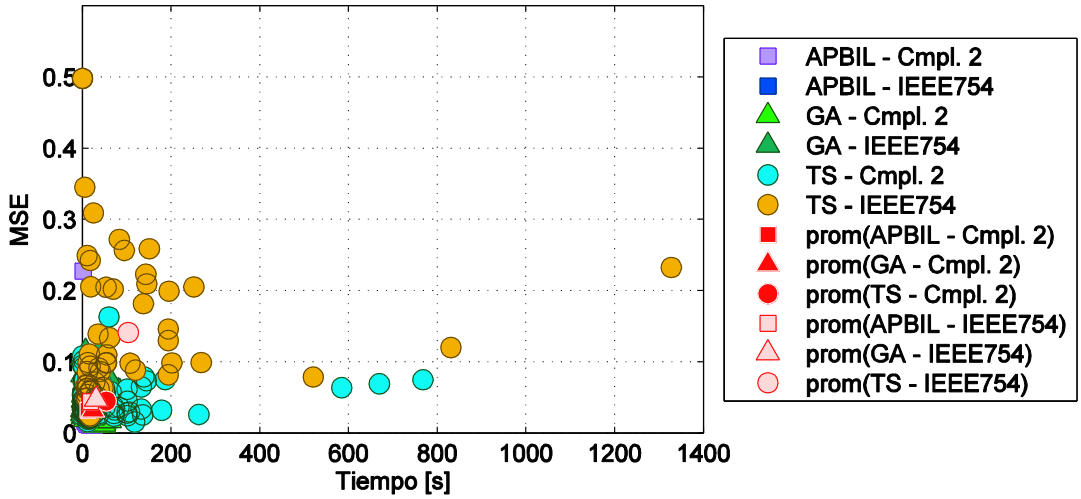
- El algoritmo de TS es el más inestable, especialmente al asociarse con la representación de *punto flotante – IEEE 754*, sus resultados fluctúan mucho en cuanto a la calidad de las soluciones, y sus tiempos de respuesta, aunque variados, pueden superar los 20 minutos. En relación a su asociación con la representación de *punto fijo – complemento a dos* se pueden esperar resultados competentes, especialmente si las condiciones refieren el uso de pocos *tabs*, asociado a bajas precisiones.

- En relación a la calidad, las soluciones proporcionadas por los algoritmos APBIL y GA son estables y comparables entre sí (basta con observar el nivel de los valores promedio de sus series de datos). Pero, respecto a los tiempos de convergencia, el algoritmo APBIL presenta datos más concentrados, que no superan los 45 segundos (con desviaciones estándar de 13.34 y 18.07 segundos, para las representaciones de punto fijo y punto flotante, respectivamente), a diferencia del GA, cuyos datos se extienden hasta el minuto y medio (con desviaciones estándar de 24.15 y 31.11 segundos, para las representaciones de punto fijo y punto flotante, respectivamente).

- Además, al observar el comportamiento de las series, discriminado por tipo de representación numérica, se evidencia que la representación de punto flotante (según el estándar IEEE 754), en comparación con la representación de punto fijo (complemento a

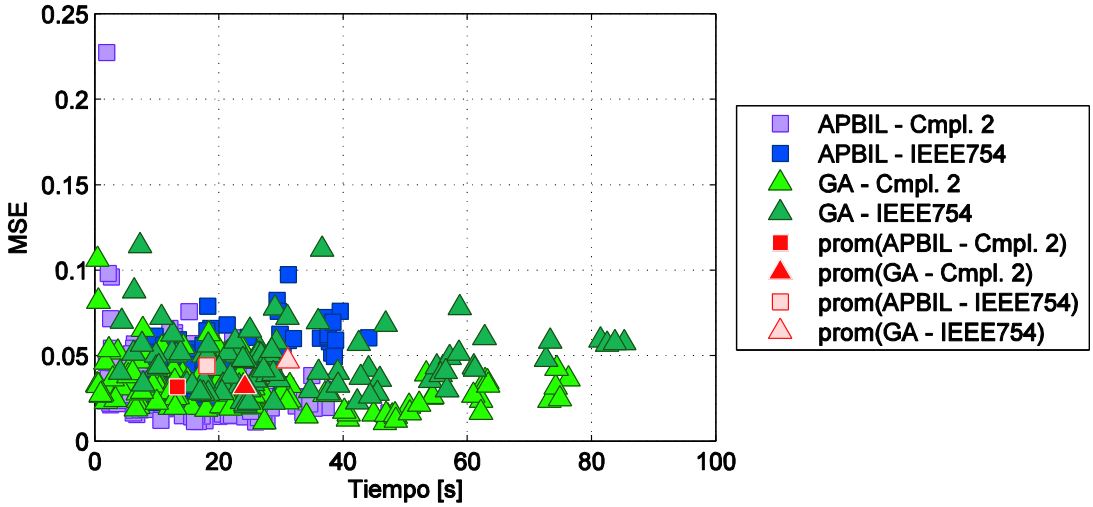
dos), proporciona soluciones de menor calidad, que adicionalmente, tardan más en ser obtenidas.

Figura 3-5: Datos de las pruebas de desempeño de los algoritmos APBIL, GA y TS.



Nombre de la fuente: Elaboración propia

Figura 3-6: Datos de las pruebas de desempeño de los algoritmos APBIL y GA.



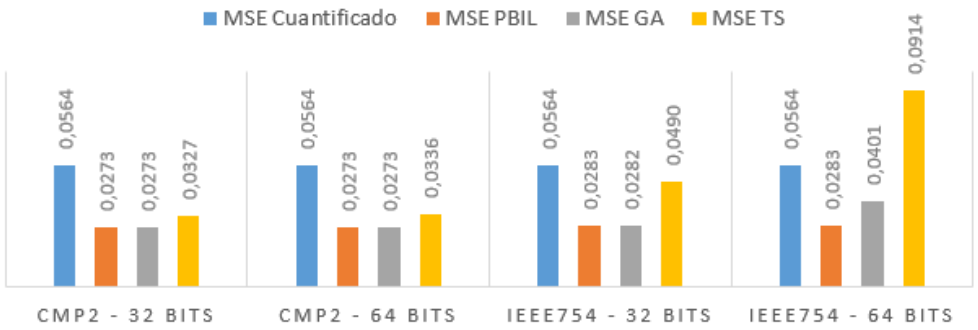
Nombre de la fuente: Elaboración propia

- Comparación de los mejores resultados obtenidos

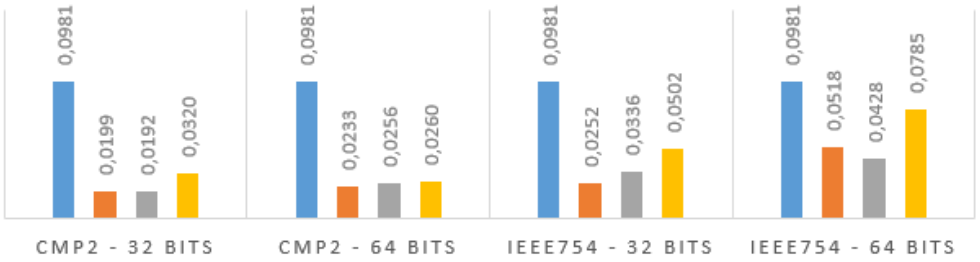
A continuación, se reportan los mejores resultados de las pruebas anteriores, que contemplan diseños FIR e IIR, diversos tipos de respuestas (LP, HP, PB, SB y MB de tres bandas) y diferentes valores para el orden (de 2 a 6 para los diseños IIR y de 6 a 9 para los diseños FIR). En resumen,

de los casos contemplados, el algoritmo APBIL superó al diseño tradicional el 87.13% de las veces, el GA en un 77.44% y el algoritmo TS el 55.83%. Para todos los casos, la primer barra (azul) representa el MSE del diseño tradicional cuantificado, la segunda barra (roja) corresponde al MSE del diseño APBIL, la tercera (gris) al MSE del diseño del GA y la cuarta barra (amarilla) se relaciona con el diseño del algoritmo TS.

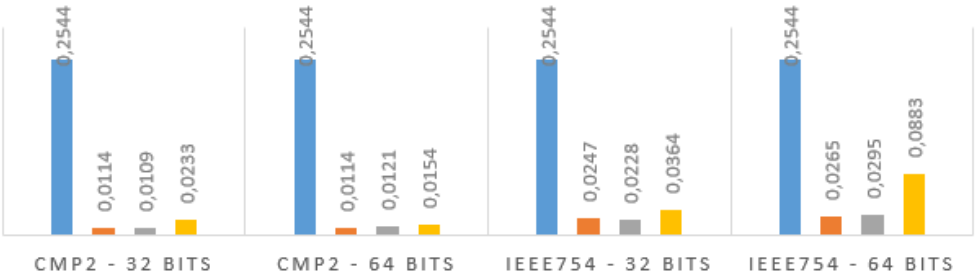
Butterworth, HP, de segundo orden



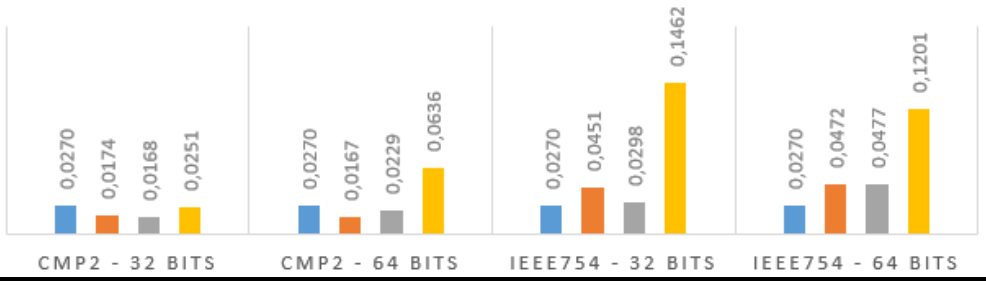
Chebyshev tipo I, PB, de tercer orden



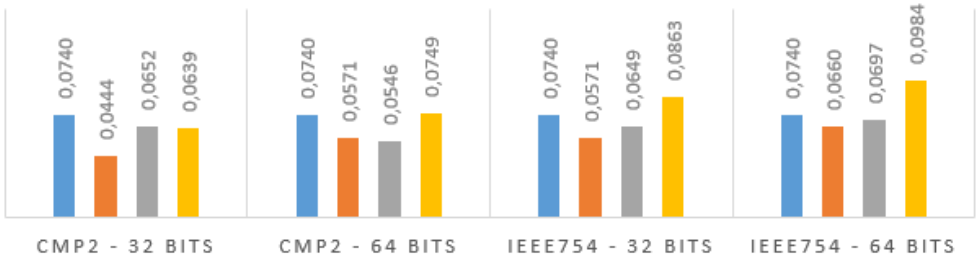
Chebyshev tipo II, LP, de cuarto orden



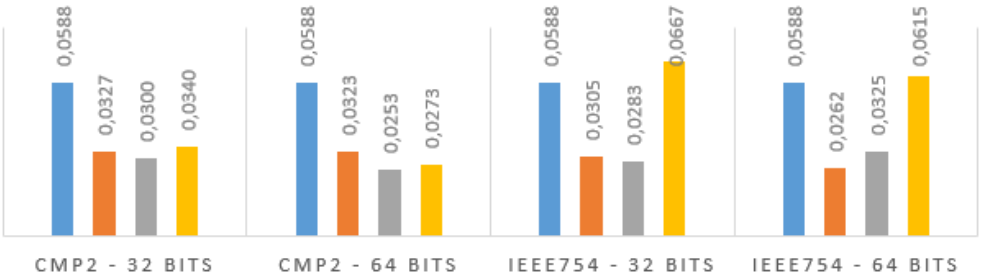
Elíptico o de Causer, SB, de quinto orden



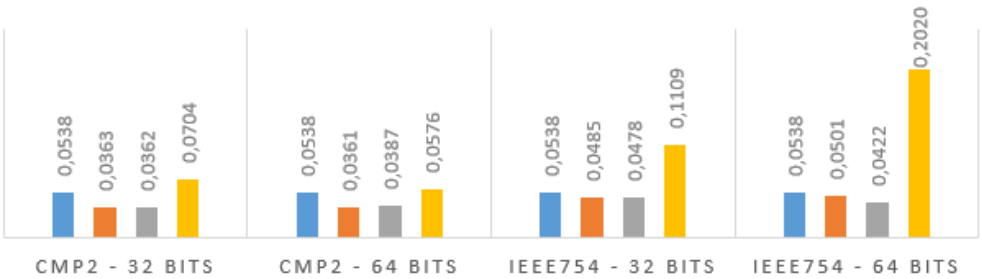
Yelewalk (método de mínimos cuadrados), MB de 3 bandas, de sexto orden



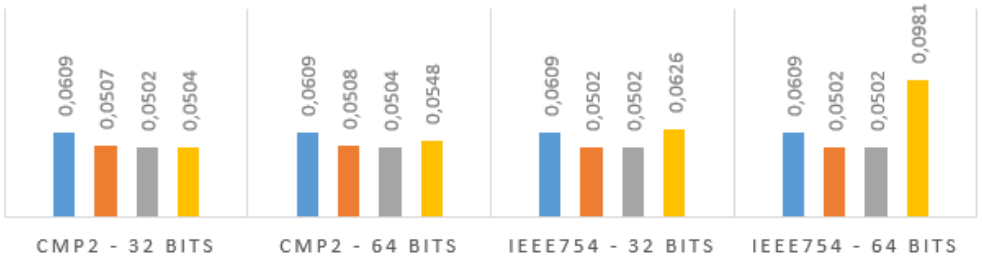
Método de Enventanado (fase lineal), ventana Gaussiana, HP, de sexto orden



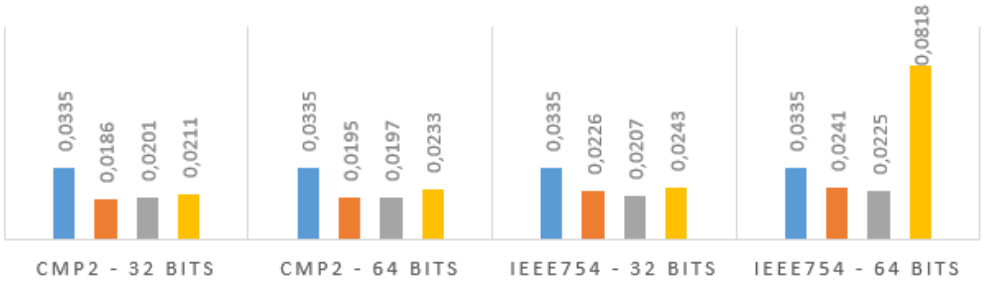
Método de Muestreo en Frecuencia, ventana Rectangular, SB, de octavo orden



Método de Mínimos Cuadrados (fase lineal), simétrico, PB, de séptimo orden

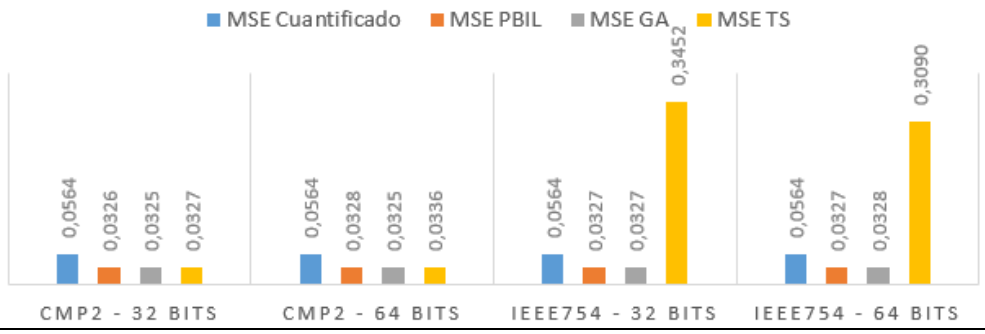


Algoritmo de Parks-McClellan, anti simétrico, HP, de noveno orden

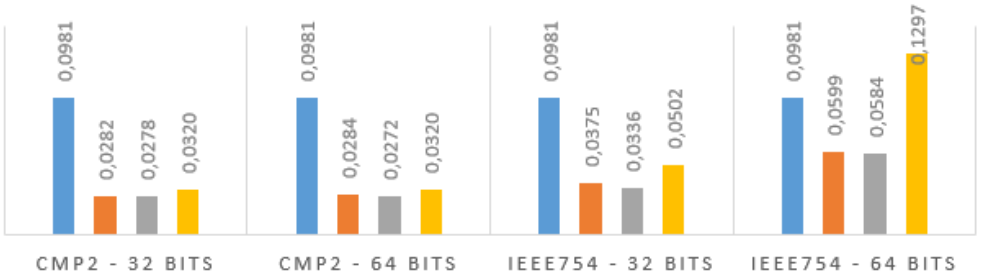


Como muestra adicional de la efectividad de las metodologías propuestas, se han diseñado un conjunto de filtros FIR bajo las condiciones contempladas en los diseños IIR de las pruebas anteriores, es decir, se conservó el mismo número de *tabs* de los diseños IIR, al igual que su precisión y representación numérica. Como resultado, se obtuvieron filtros FIR que superan en calidad a los filtros IIR obtenidos por medio de los métodos tradicionales, en otras palabras, se aproximan más al filtro ideal a pesar de que el número de *tabs* es igual. El 82.50% de los diseños FIR obtenidos con el algoritmo APBIL superaron los diseños IIR tradicionales, el GA logró hacerlo en un 81.48% de las veces y el algoritmo TS con el 45.45%. A continuación se reportan los mejores resultados de estas pruebas.

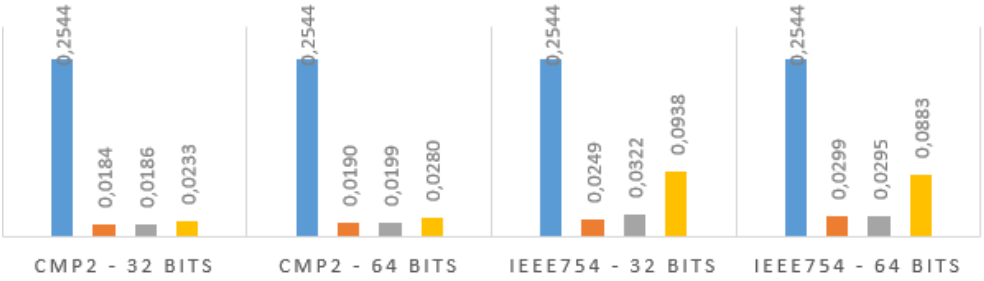
Butterworth, HP, segundo orden (seis *tabs*)



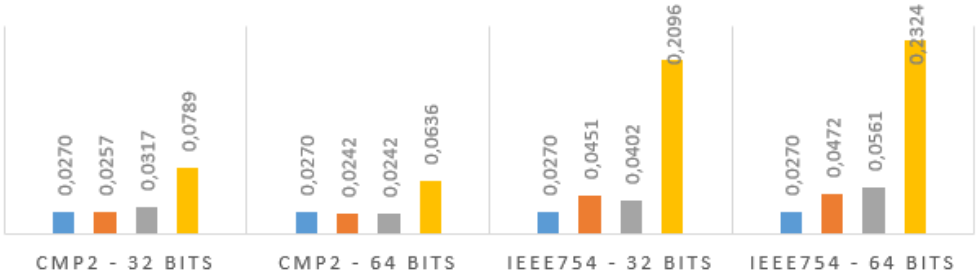
Chebyshev I, PB, tercer orden (14 *tabs*)



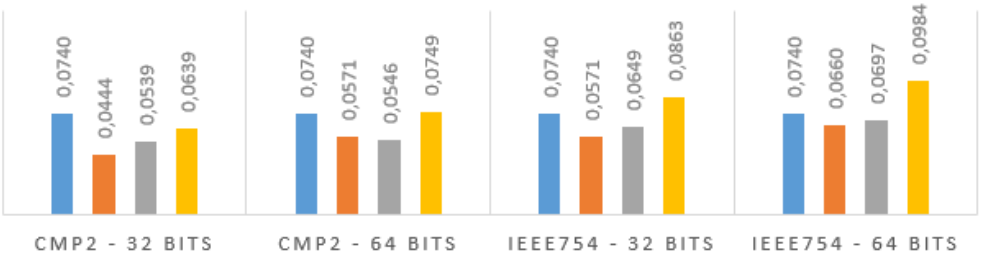
Chebyshev II, LP, cuarto orden (10 *tabs*)



Elíptico o de Causer, SB, quinto orden (22 *tabs*)



Yulewalk (mínimos cuadrados), MB de 3 bandas, sexto orden (14 tabs)

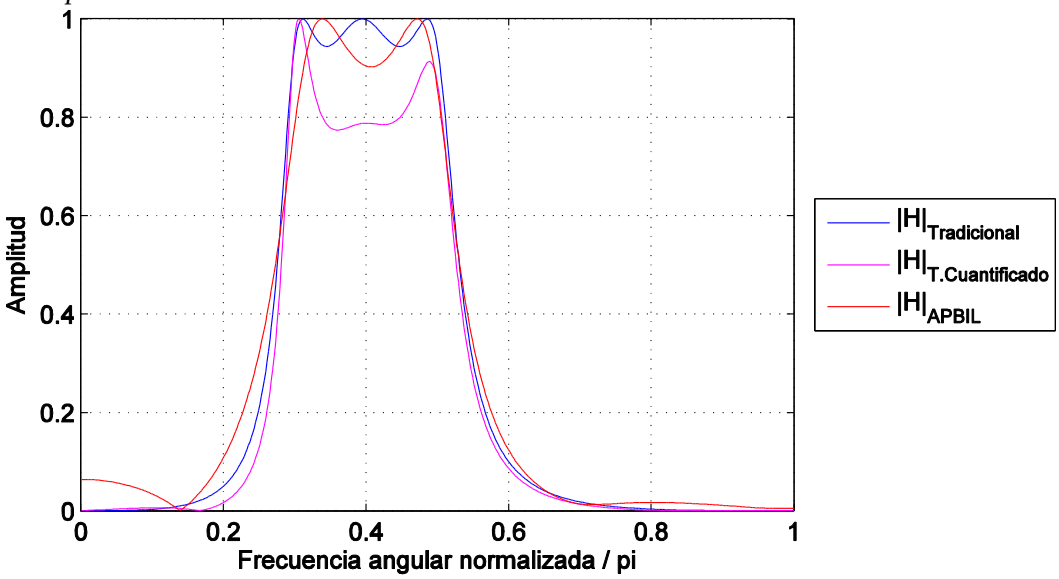


- Ejemplo 1: diseño de un filtro IIR, de bajo orden y precisión limitada

En la Figura 3-7 se presentan tres Espectros de Frecuencia normalizados, que corresponden a: un filtro Chebyshev tipo I, PB, de tercer orden, con amplitud de rizado de 0.5 dB, sin cuantificar (que desde ahora será referido como el filtro tradicional), en segundo lugar, el mismo diseño pero cuantificado, con una precisión de $k = 8$ bits por *tab* y representación numérica de *punto fijo – complemento a dos*, y por último, un diseño IIR obtenido con la herramienta de optimización APBIL, que posee el mismo número de *tabs*, precisión y representación numérica que el filtro tradicional cuantificado y corresponde a una aproximación al filtro tradicional (y no al ideal, como si ocurría en las pruebas anteriores). Se observa como el proceso de cuantificación, efectuado al final de proceso de diseño, degrada la respuesta original del diseño tradicional, mientras que, el diseño propuesto por el algoritmo APBIL, el cual contempla la cuantificación desde un comienzo, brinda una mejor aproximación a la respuesta deseada, que en este caso es el filtro tradicional original. Lo anterior se verifica con los errores calculados para las alternativas: el diseño tradicional cuantificado obtuvo un $MSE = 0.005197$, contra un $MSE = 0.002211$ del diseño APBIL. Por otro lado, en la

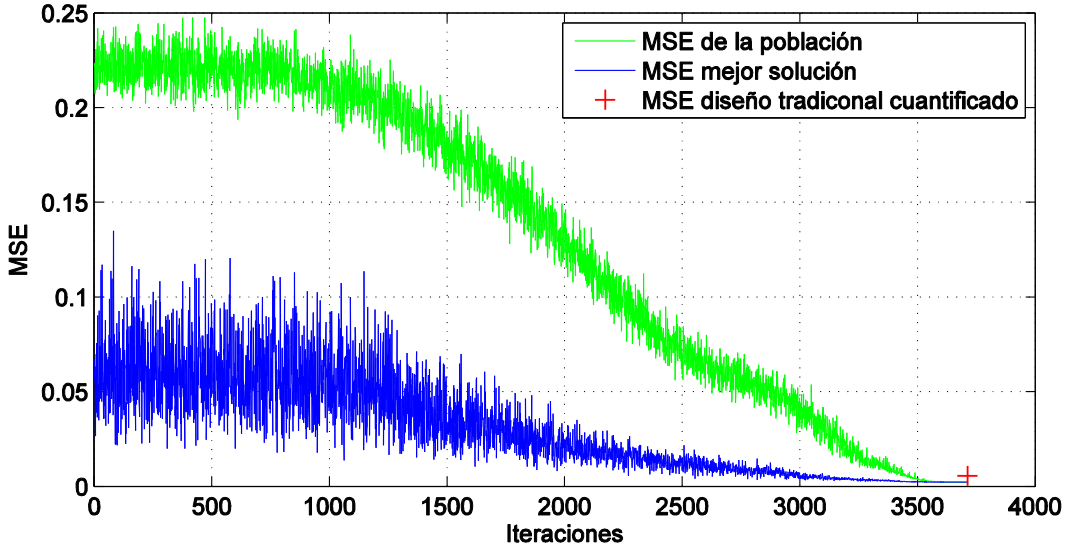
Figura 3-8 se muestran la evolución del MSE de la mejor solución calculada por el algoritmo APBIL y del MSE promedio de la población, dentro del proceso iterativo.

Figura 3-7: Diseño APBIL vs. Diseño Chebyshev tipo I, PB, tercer orden, 8 bits por *tab*, *punto fijo – complemento a dos*.



Nombre de la fuente: Elaboración propia

Figura 3-8: Evolución del MSE del algoritmo APBIL.



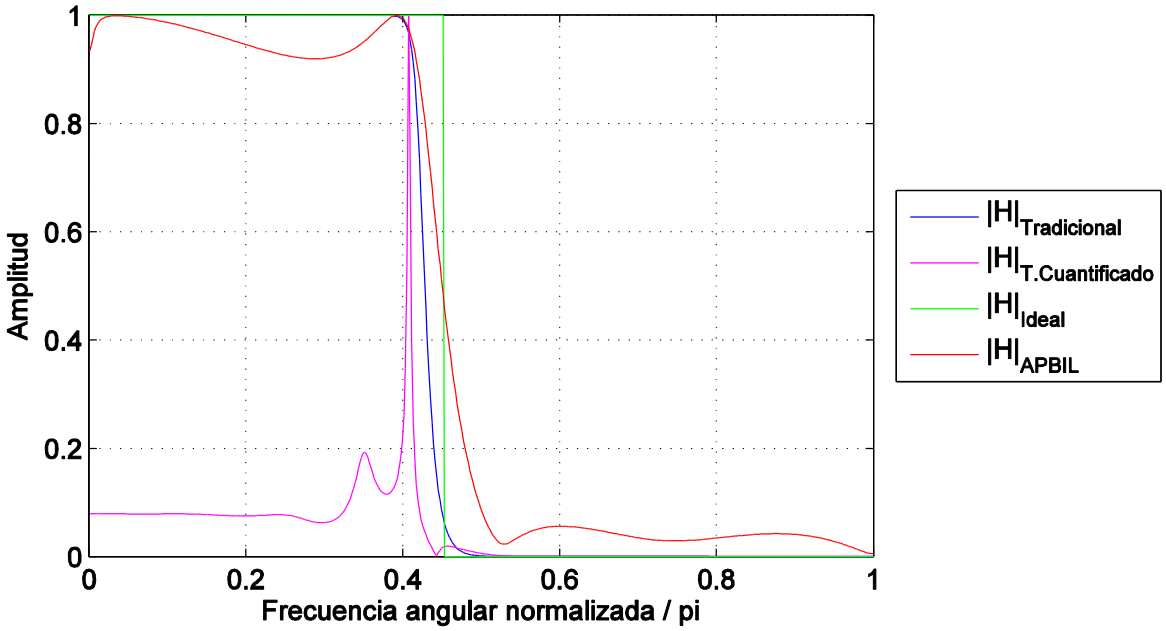
Nombre de la fuente: Elaboración propia

- Ejemplo 2: ahorro en bloques de memoria

Existen casos en los cuales se pueden obtener mejores diseños, con un número menor de coeficientes, cuando se utilizan las metodologías de diseño propuestas, basadas en las herramientas de optimización APBIL, GA y TS (destacándose el uso del algoritmo APBIL), que cuando se empelan las metodologías tradicionales. La obtención de mejores diseños con un número menor de coeficientes (o memorias) implica que las metodologías de diseño propuestas (especialmente aquella que implementa el algoritmo APBIL), son capaces de reducir los costos de implementación de los sistemas requeridos, que llevado a un marco de producción en masa de tecnologías digitales de alto consumo, puede significar un ahorro significativo.

Un ejemplo de ello es el filtro digital IIR, LP; presentado en la Figura 3-9, donde el algoritmo ABPIL implementa un diseño de séptimo orden (16 *tabs*), obteniendo mejores resultados que al empelar un diseño Butterworth de orden 28 (56 *tabs*). Además, se observa cómo por causa de la cuantificación, para una precisión de $k = 16$ bits por *tab* y representación de *punto fijo – complemento a dos*, el diseño tradicional se aleja del original y se degrada a tal punto de que deja de ser selectivo en frecuencia. El error respecto al diseño ideal, obtenido por el diseño tradicional cuantificado es de 0.050018, mientras el error del diseño APBIL es igual a 0.008696.

Figura 3-9: Diseño APBIL, de séptimo orden, vs. Diseño Butterworth de orden 28, LP, 16 bits por *tab, punto fijo – complemento a dos*.



Nombre de la fuente: Elaboración propia

Es necesario aclarar que todas las pruebas fueron realizadas en el Aplicativo desarrollado para Windows, a través de un PC con 12 Gbytes de memoria RAM y un procesar Core i7.

4. Conclusiones y recomendaciones

4.1 Conclusiones

Se han descrito y probado tres herramientas de optimización metaheurística para el diseño de filtros digitales lineales e invariantes en el tiempo, FIR e IIR, que abordan el proceso de cuantificación desde el principio y a lo largo del proceso de diseño. Los resultados de las simulaciones son prometedores y muestran un buen desempeño de los filtros obtenidos, en términos del Error Medio Cuadrático (MSE) entre las respuestas en frecuencia deseada y la obtenida a partir del filtro optimizado. En particular, las herramientas APBIL y GA demuestran tener la capacidad de proporcionar resultados que superan en calidad a la respuesta entregada por las metodologías tradicionales, especialmente, cuando se trata de la implementación de filtros digitales con número reducido de coeficientes (*tabs*) y/o baja precisión numérica.

Una de las ventajas de las metodologías de diseño de filtros digitales propuestas, consiste en que los coeficientes del filtro son representados por su codificación numérica durante el proceso de optimización. Permitiendo, a la vez, que la etapa de cuantificación sea contemplado a largo de todo el proceso de diseño, y no al final, como ocurre en otras metodologías, incluidas las metodologías tradicionales, en la cuales, dicha acción puede incrementar el error de la aproximación (ver Figura 3-7 y Figura 3-9).

Los procesos de sintonización efectuados en las herramientas de optimización metaheurística buscan el desempeño óptimo del algoritmo (entendido como el balance adecuado entre tiempo de convergencia y calidad de la respuesta), explotan la capacidad de obtener buenos resultados (o malos) y la confianza que se pude tener en ellos para poder obtener datos semejantes cada vez. Los procesos de sintonización favorecieron el desempeño de cada algoritmo. El mejoramiento de las soluciones, la reducción de los tiempos de convergencia y la disminución de la variabilidad de los resultados fueron evidentes, al finalizar el proceso y también paso a paso con cada nuevo ajuste realizado.

El plan de sintonización desarrollado e implementado en las tres herramientas de optimización metaheurística fue efectivo. Por lo cual, se dedujo un Esquema General de Sintonización (ver página 35), que puede servir de referencia al lector de modo que sea aplicado como plan de sintonización de cualquier otro algoritmo de optimización metaheurística similar.

Dentro del contexto del diseño de filtros digitales lineales e invariantes en el tiempo, La sintonización recomendada para los algoritmos de optimización GA, TS y APBIL, es la mostrada en las Tablas 2-9, 2-11 y 2-15, respectivamente. Adicionalmente, existen valores para los parámetros de los algoritmos, identificados a lo largo del proceso de diseño, que si bien son diferentes a la sintonización recomendada, están en la capacidad de favorecer el desempeño del algoritmo bajo condiciones diferentes a las presentadas en la sintonización final. En las Tablas 2-7 y 2-8 se presentan las alternativas a la sintonización del GA y en las Tablas 2-13 y 2-14 para el algoritmo APBIL. Valores o combinaciones diferentes a las recomendadas no garantizan el rendimiento óptimo del algoritmo, ni el hallazgo de buenas soluciones.

De entre las metodologías de diseño de filtros digitales propuestas, bajo la sintonización recomendada (ver las Tablas 2-9, 2-11 y 2-15), aquella que implementa el algoritmo APBIL es la más rápida en respuesta y la que obtuvo los mejores resultados en calidad. De los casos evaluados, en los cuales se igualaron las condiciones del tipo de respuesta, del número *tabs*, precisión y representación numérica, el algoritmo APBIL superó al diseño tradicional el 87.13% de las veces, el GA lo hizo el 77.14% y el algoritmo TS en un 55.4%. Respecto a los tiempos de convergencia, el algoritmo APBIL presenta datos más concentrados, que no superan los 45 segundos (con desviaciones estándar de 13.34 y 18.07 segundos, para las representaciones numéricas de punto fijo y punto flotante, respectivamente), a diferencia del GA, cuyos datos se extienden hasta el minuto y medio (con desviaciones estándar de 24.15 y 31.11 segundos, para las representaciones de punto fijo y punto flotante, respectivamente), y por otro lado, los tiempos de respuesta del algoritmo TS, aunque variados, pueden llegar a superar los 20 minutos (ver las Figuras 3-5 y 3-6).

Como resultado de la implementación de los algoritmos APBIL, GA y TS para el diseño de filtros digitales lineales e invariantes en el tiempo, se construyó un *software*, con una interfaz de usuario amigable, que permite realizar diseños FIR e IIR empleando cualquiera de las herramientas de optimización metaheurística propuestas. Dicho *software*, presenta las mismas ventajas que las

metodologías de diseño descritas en este documento, pues a diferencia de muchas herramientas reportadas, contempla la cuantificación y codificación desde un comienzo del proceso de diseño, y no solo al final del mismo. Adicionalmente, y de forma simultánea, permite obtener el diseño bajo las técnicas tradicionales más conocidas, proporcionando el valor de los coeficientes ideales (previos a la cuantificación) y cuantificados, que a su vez, utiliza para realizar comparaciones gráficas de los Espectros de Frecuencia y de los errores de los diseños obtenidos.

La herramienta de *software*, o aplicativo, desarrollado para el diseño de filtros digitales lineales e invariantes en el tiempo, puede ser distribuida a diferentes profesionales, que estando interesados en obtener el diseño de un filtro digital para implementarlo en su campo respectivo, sin ser expertos en la temática, pueden obtener por sí mismos un diseño acorde a sus necesidades y con la mejor calidad posible. Gracias a que la interfaz de usuario desarrollada es de fácil manejo y contempla como valores por defecto los sugeridos por los resultados de los exhaustivos procesos de sintonización efectuados en los algoritmos de optimización metaheurística propuestos.

Existen casos en los cuales se pueden obtener mejores diseños, con un número menor de coeficientes, cuando se utilizan las metodologías propuestas, basadas en las herramientas de optimización APBIL, GA y TS (destacándose el uso del algoritmo APBIL), que cuando se empelan las metodologías tradicionales (ver Ejemplo 2: ahorro en bloques de memoria, página 100). La obtención de mejores diseños con un número menor de coeficientes (o memorias) implica que las metodologías de diseño propuestas (especialmente aquella que implementa el algoritmo APBIL), son capaces de reducir los costos de implementación de los sistemas requeridos, que llevado a un marco de producción en masa de tecnologías digitales de alto consumo, puede significar un ahorro económico significativo.

Los resultados (mostrados en la página 100) demuestran que las metodologías propuestas pueden lograr implementaciones FIR del mismo orden que una IIR, pero de mejor desempeño, desvirtuando lo que tradicionalmente se afirma de que para una misma aplicación, las implementaciones IIR son más económicas, pero más peligrosas (por aquello de la inestabilidad), y que las FIR son más seguras, pero son más cotosas.

El aplicativo cuenta con dos elementos adicionales, la inserción de la solución anterior como semilla en un nuevo proceso de diseño y la comparación no uniforme del MSE, que pueden mejorar los resultados obtenidos por los algoritmos de optimización metaheurística. Emplear la

solución anterior para generar a partir de ella un nuevo diseño, permite actuar a los algoritmos como si estuvieran conectados en cascada para producir en conjunto una solución, que en términos del MSE pude ser mejor que la solución anterior (como se muestra en la Tabla 2-18). Por otro lado, la comparación no uniforme del MSE, permite asignar un peso o porcentaje de importancia a la(s) zona(s) de transición en la evaluación del error, permitiendo acentuar en el proceso de diseño la importancia de las características de la(s) banda(s) de transición, y forma indirecta, la importancia de las características de las bandas de paso y rechazo, en otras palabras, permite manipular la importancia de la aproximación e inclinar la balanza hacia la(s) banda(s) de transición o hacia las demás.

4.2 Recomendaciones

Aunque los resultados de las pruebas de desempeño demuestran que las metodologías de diseño propuestas, basadas en los algoritmos APBIL, GA y TS, están en capacidad de brindar soluciones de calidad, que más allá de ser competitivas, alcanzan a superar los diseños arrojados por las metodologías tradicionales, es una ventaja que se va reduciendo conforme se incrementa el número de coeficientes del sistema y/o se aumenta la precisión numérica. Como se explicó en varias ocasiones, incrementar dichos valores, que en síntesis es aumentar el número de bits de la representación de la solución, representa para los algoritmos de optimización metaheurística sumar variables al problema de búsqueda de la solución óptima, aumentando su complejidad, mientras que, por el contrario, para las metodologías de diseño tradicional significa mejorar la aproximación. Una mejora a las metodologías propuestas, en especial para aquella que implementa el algoritmo APBIL, el cual destacó por su desempeño, es encontrar un método que potencie sus ventajas aun cuando el tamaño del problema se incremente.

Otro camino para mejorar los resultados obtenidos es perfeccionar la manera en que se realiza la comparación de las respuestas en frecuencia. Actualmente, la comparación es hecha punto a punto, entre puntos que corresponden a la misma ubicación vertical, es decir a la misma muestra de frecuencia, con una distribución homogénea entre todas las muestras de frecuencia. Esto pone en desventaja la aproximación a las bandas de transición con respecto a las bandas de paso y rechazo, pues son menos los puntos empelados para realizar las comparaciones respectivas. Con intención de atacar dicha desventaja, al aplicativo de diseño de filtros digitales desarrollado se le agregó la opción de la evaluación no uniforme del MSE (a través de la Ecuación (2-15)), pero, la evaluación aún se realiza para una distribución uniforme de las muestras de frecuencia. Una mejora a esta

situación es encontrar la manera de realizar una comparación de los Espectros de Frecuencia con una distribución no uniforme de las muestras de frecuencia, en donde se aumente la densidad de puntos en la(s) banda(s) en las que se desea máxima precisión en la aproximación.

Bibliografía

[1] A. Albiol, V. Naranjo, and J. Prades, *Tratamiento Digital de la Señal Teoría y Aplicaciones*. Valencia, España, 1999.

[2] A. V. Oppenheim and R. W. Schafer, *Tratamiento de Señales en Tiempo Discreto*, 3rd ed. 2011.

[3] J. G. Proakis, *Tratamiento digital de señales*, 3rd ed. Madrid, Esp.: Prince Hall Latinoamericana, 2005.

[4] S. W. Smith, *the Scientist and Engineer's Guide to Digital Signal Processing*, 2nd ed. San Diego, California: California Technical Publishing.

[5] K. Ortega and F. Bolanos, "A Population-Based Incremental Learning Tool for Finite Impulse Response (FIR) Filters Design." In Proceedings on the International Conference on Artificial Intelligence (ICAI) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[6] J. G. Proakis and V. K. Ingle, *Digital Signal Processing using MATLAB*, 3rd ed. Global Engineering, 2012.

[7] Institute of Electrical and Electronics Engineers, *"IEEE Standard for Binary Floating-Point Arithmetic,"* 1987.

[8] J. R. Camarillo-Peñaranda *et al., "Reconfiguration of photovoltaic arrays based on genetic algorithm Reconfiguración de arreglos fotovoltaicos basada en algoritmo genético," Rev. Fac. Ing. Univ. Antioquia*, vol. 75, pp. 95–107, 2015.

[9] D. J. Krusienski and W. K. Jenkins, *"Particle swarm optimization for adaptive IIR filter structures,"* Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), 2004, pp. 965-970 Vol.1.

[10] A. Kalinli and N. Karaboga, *"A new method for adaptive IIR filter design based on tabu search algorithm,"* AEU - Int. J. Electron. Commun., 2005.

[11] N. Karaboga*, "A new design method based on artificial bee colony algorithm for digital IIR filters,"* J. Franklin Inst., 2009.

[12] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, *"Filter modeling using gravitational search algorithm,"* Eng. Appl. Artif. Intell., 2011.

[13] M. Sharifi and H. Mojallali, *"Design of IIR Digital Filter using modified chaotic orthogonal imperialist competitive algorithm,"* 2013.

[14] A. Kalinli and N. Karaboga, *"Artificial immune algorithm for IIR filter design,"* Engineering Applications of Artificial Intelligence. 2005.

[15] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, *"Optimal Design of Digital IIR Filters by Using Hybrid Taguchi Genetic Algorithm,"* IEEE Trans. Ind. Electron., vol. 53, no. 3, 2006.

[16] B. Luitel and G. K. Venayagamoorthy, *"Differential Evolution Particle Swarm Optimization for Digital Filter Design,"* in IEEE Congress on Evolutionary Computation (CEC 2008), 2008.

[17] N. Karaboga, A. Kalinli, and D. Karaboga, *"Designing digital IIR filters using ant colony optimisation algorithm,"* Eng. Appl. Artif. Intell., vol. 17, no. 3, pp. 301–309, Apr. 2004.

[18] K. A. Folly and G. K. Venayagamoorthy, "Power system controller design using multi-population PBIL," in IEEE Symposium on Computational Intelligence Applications in Smart Grid, CIASG, 2013.

[19] L. J. Fan, B. Li, Z. Q. Zhuang, and Z. Q. Fu, *"An approach for dynamic hardware /software partitioning based on DPBIL,"* in Proceedings - Third International Conference on Natural Computation, ICNC 2007, 2007.

[20] F. Bolanos, J. E. Aedo, F. Rivera, and N. Bagherzadeh, *"Mapping and Scheduling in Heterogeneous NoC through Population-Based Incremental Learning,"* J. Univers. Comput. Sci., vol. 6, no. 6, 2012.

[21] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. New York: Jhon Wiley & Sons, 1998.

[22] Y. Yu and Y. Xinjie, *"Cooperative Coevolutionary Genetic Algorithm for Digital IIR Filter Design,"* IEEE Trans. Ind. Electron., vol. 54, no. 3, pp. 1311–1318, 2007.

[23] F. Bolaños and F. A. Ramírez, *"Aprendizaje Incremental Basado en Población: Una herramienta estocástica para optimización,"* Revista Avances en Inteniería Eléctrica, Medellín, pp. 68–73, Nov-2014.

[24] F. Bolanos, J. E. Aedo, and F. Rivera, *"Comparison of Learning Rules for Adaptive Popultation-Based Incremental Learning Algorithms."* In *Proceedings on the International Conference on Artificial Intelligence (ICAI)* (p. 1). The Steering Committee of The World

Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).